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

KỸ THUẬT DÀNH CHO KIẾN TRÚC VÀ THIẾT KẾ

TS. Huỳnh Hữu Nghĩa

huynhhuunghia@iuh.edu.vn

Nội dung:

- Tổng quan
- Các bước thiết kế
- Xác định các mục tiêu kiến trúc
- Các kịch bản chính
- Tổng quan ứng dụng
- Các vấn đề chính
- Các giải pháp ứng viên
- Đánh giá lại kiến trúc
- Trình bày và trao đổi kiến trúc

Tổng quan

- Giúp đưa ra những quyết định quan trọng:
  - ✓ Các thuộc tính chất lượng,
  - ✓ Kiểu kiến trúc,
  - ✓ Kiểu ứng dụng,
  - ✓ Các công nghệ, và
  - ✓ Những quyết định phát triển.

Tổng quan

- Tùy thuộc vào hướng tiếp cận của tổ chức đối với phát triển phần mềm, có thể xem xét lại kiến trúc nhiều lần trong suốt thời gian thực hiện dự án.

- Có thể sử dụng kỹ thuật này để tinh chỉnh kiến trúc, xây dựng dựa trên những gì được biết trong thời gian can thiệp vào việc đột phá, tạo mẫu, và phát triển thực tế.

Đầu vào và Đầu ra

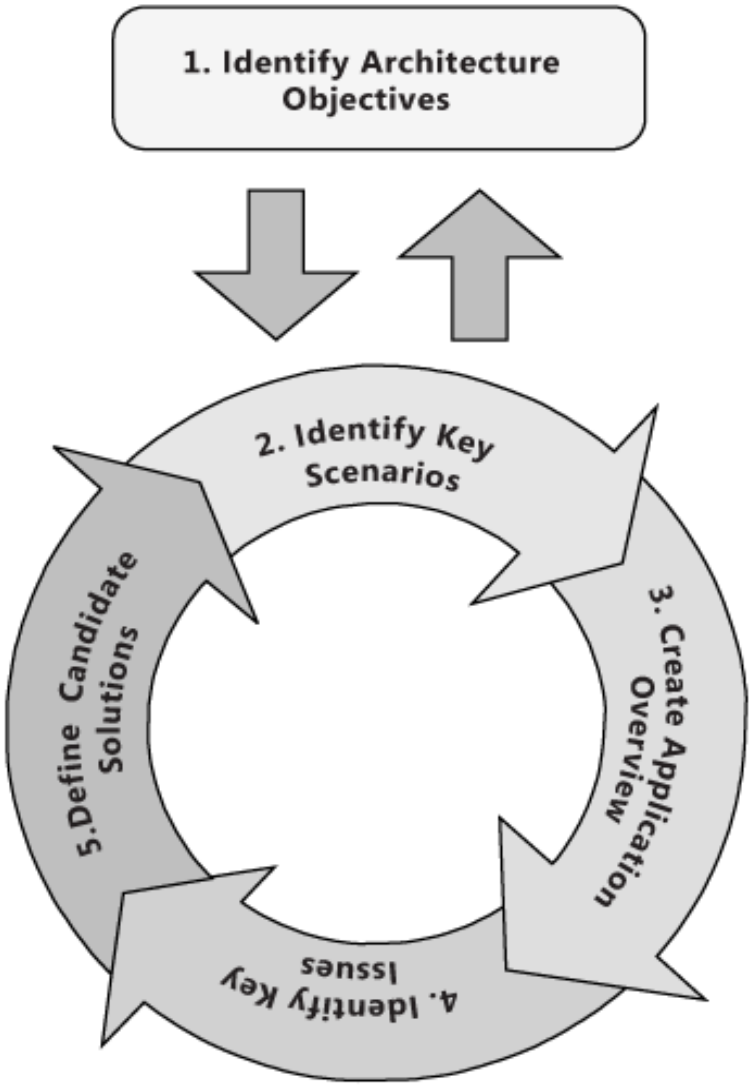
- Đầu vào cho thiết kế có thể giúp chính thức hóa các yêu cầu và các ràng buộc mà kiến trúc phải phù hợp.

- Các đầu vào thông thường là các kịch bản use case, các yêu cầu chức năng, yêu cầu phi chức năng (gồm các thuộc tính chất lượng như: *hiệu quả*, *bảo mật*, *độ tin cậy* và những cái khác), yêu cầu công nghệ, môi trường phát triển mục tiêu, và các ràng buộc khác.

Đầu vào và Đầu ra

- Trong quá trình thiết kế, sẽ tạo ra danh sách các use case có ý nghĩa về mặt kiến trúc, những vấn đề kiến trúc yêu cầu chú ý đặc biệt, và các giải pháp kiến trúc ứng viên đáp ứng những yêu cầu và ràng buộc được xác định trong quá trình thiết kế.

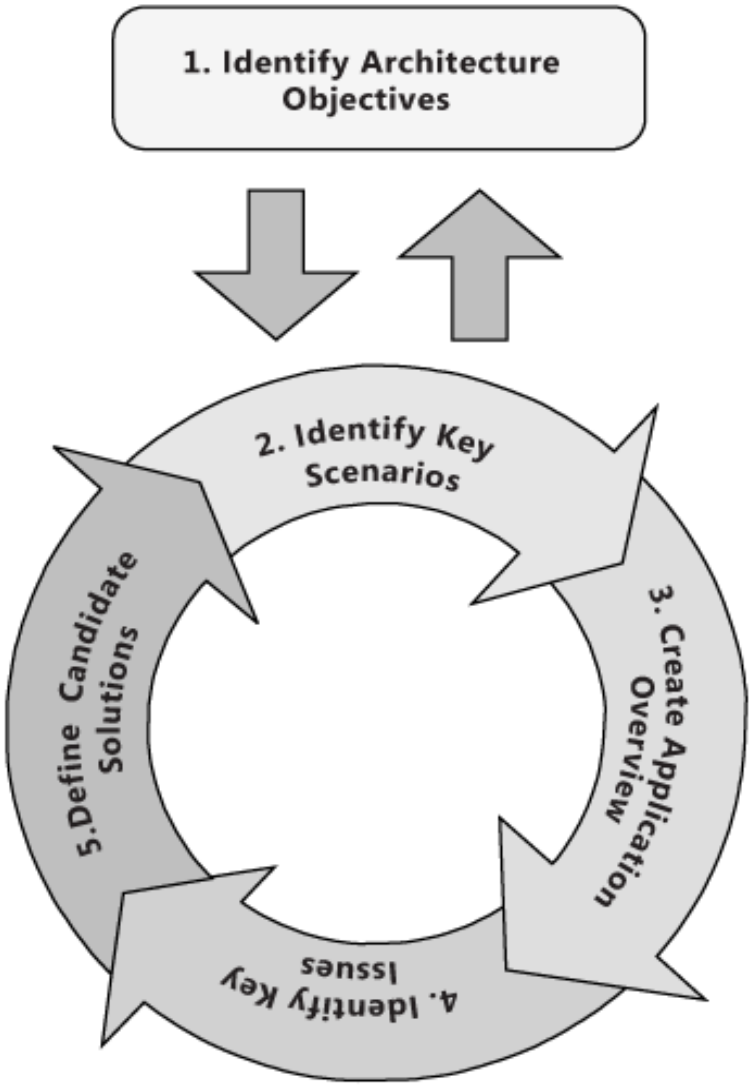
Các bước thiết kế



1. Identify Architecture Objectives
(Xác định các mục tiêu kiến trúc)

(Các mục tiêu rõ ràng giúp bạn tập trung vào kiến trúc và giải quyết các vấn đề chính trong thiết kế. Các mục tiêu chính xác giúp xác định thời điểm đã hoàn thành giai đoạn hiện tại, và khi đã sẵn sàng chuyển sang bước tiếp theo.)

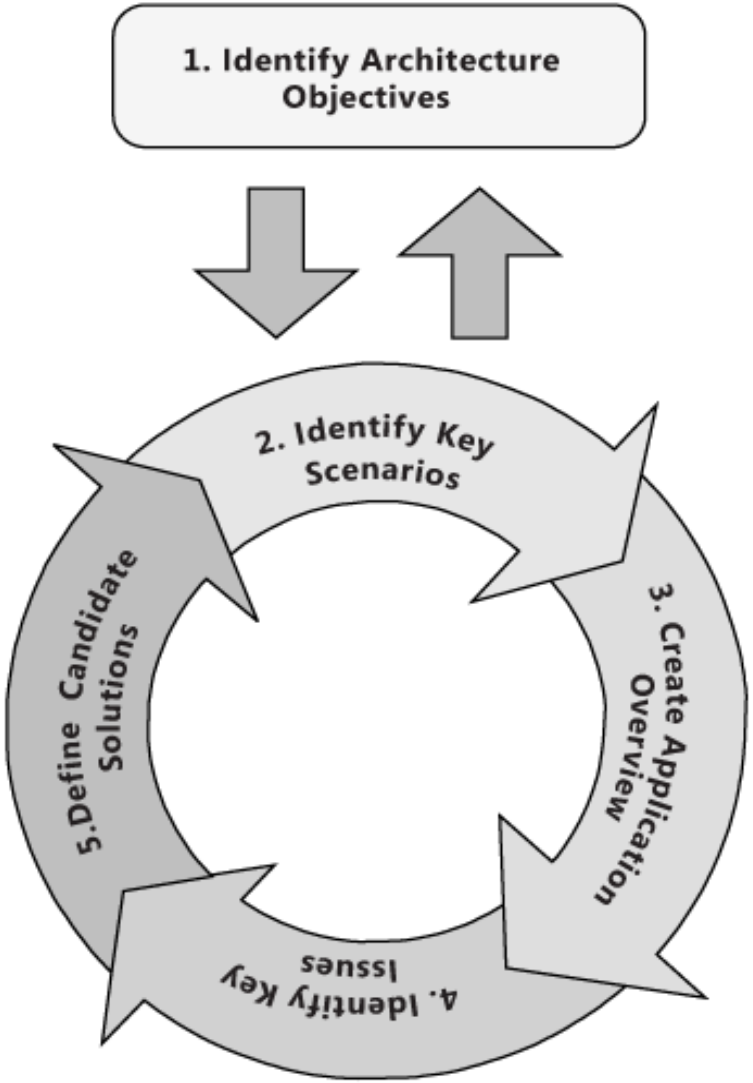
Các bước thiết kế



2. Identify Key Scenarios (Xác định các kịch bản chính)

(Sử dụng các kịch bản chính để tập trung đến thiết kế vào những gì quan trọng nhất, và đánh giá các kiến trúc ứng viên khi chúng sẵn sàng.)

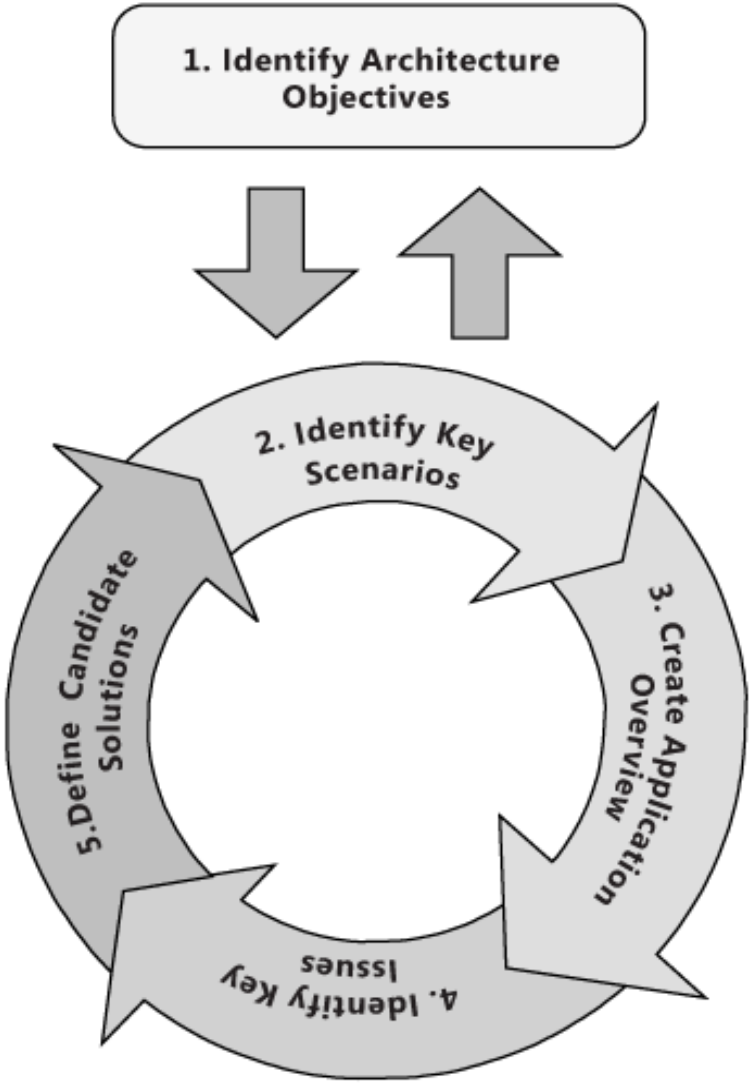
Các bước thiết kế



3. Create Application Overview
(Tạo tổng quan ứng dụng)

(Xác định kiểu ứng dụng, kiến trúc phát triển, kiểu kiến trúc, và công nghệ để kết nối thiết kế đến thế giới thực mà ứng dụng sẽ hoạt động.)

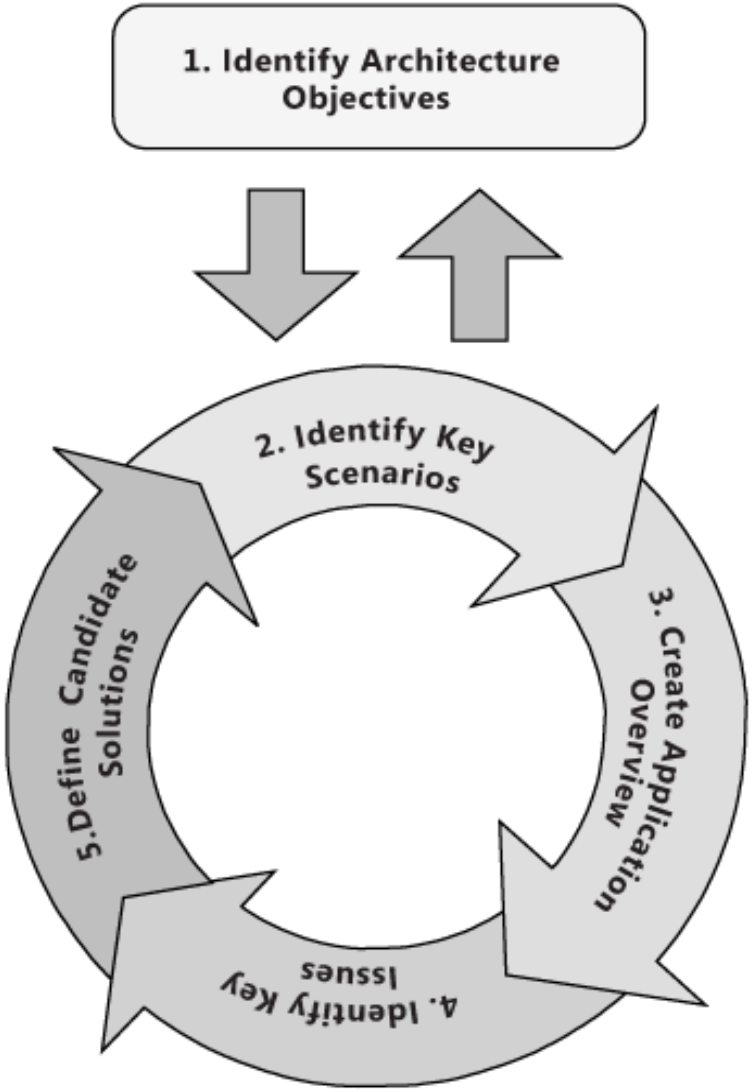
Các bước thiết kế



4. Identify Key Issues (Xác định các vấn đề chính)

(Xác định những vấn đề chính dựa trên các thuộc tính chất lượng và các mối quan tâm xuyên suốt. Đây là những lĩnh vực mà những sai lầm thường xuyên xảy ra khi thiết kế ứng dụng.)

Các bước thiết kế



5. Define Candidate Solutions (Xác định các giải pháp ứng viên)

(Việc tạo một đột biến hoặc nguyên mẫu kiến trúc nó phát triển và cải tiến giải pháp và đánh giá nó đối với các kịch bản chính, vấn đề, và các ràng buộc phát triển trước khi bắt đầu việc lặp tiếp theo của kiến trúc.)

Xác định mục tiêu kiến trúc

Các mục tiêu kiến trúc là những mục tiêu và ràng buộc chia sẽ kiến trúc và quy trình thiết kế, phạm vi thực hiện, và giúp xác định khi nào bạn kết thúc.

Xác định mục tiêu kiến trúc

Các điểm chính khi xác định mục tiêu kiến trúc:

- **Xác định các mục tiêu kiến trúc ngay khi bắt đầu.**

  Thời gian chi tiêu cho mỗi giai đoạn kiến trúc và thiết kế sẽ phụ thuộc vào những mục tiêu này. Ví dụ, bạn có đang xây dựng một nguyên mẫu, kiểm tra hướng tiềm năng, hay bắt tay vào một quy trình kiến trúc lâu dài cho một ứng dụng mới hay không?

Xác định mục tiêu kiến trúc

Các điểm chính khi xác định mục tiêu kiến trúc:

- **Xác định ai sẽ tiêu dùng kiến trúc.** *Xác định nếu thiết kế sẽ được sử dụng bởi những kiến trúc sư khác, hay cung cấp cho các nhà phát triển và các người thử nghiệm, nhân viên vận hành, và quản lý. Xem xét các nhu cầu và trải nghiệm của khách hàng để làm cho kết quả thiết kế dễ dàng tiếp cận hơn.*

Xác định mục tiêu kiến trúc

Các điểm chính khi xác định mục tiêu kiến trúc:

- **Xác định các ràng buộc**. *Hiểu các lựa chọn và ràng buộc công nghệ, các ràng buộc hạn chế, và các ràng buộc phát triển. Hiểu các ràng buộc ngay từ đầu để không bị lãng phí thời gian hay gặp phải những bất ngờ trong quá trình phát triển ứng dụng.*

Xác định các kịch bản chính

- (*Trong bối cảnh kiến trúc và thiết kế*) một use case là một mô tả của một tập các tương tác giữa hệ thống và một hay nhiều actors (người dùng hay hệ thống khác).

- Một kịch bản là một mô tả rộng và bao quát hơn về sự tương tác của người dùng với hệ thống, chứ không phải là một đường dẫn thông qua một use case.

Xác định các kịch bản chính

- Mục tiêu là xác định một vài kịch bản chính sẽ giúp đưa ra những quyết định về kiến trúc.
- Mục đích là đạt được sự cân bằng giữa các mục tiêu người dùng, nghiệp vụ, và hệ thống.
- Các kịch bản chính là những kịch bản được xem là các kịch bản quan trọng cho sự thành công của ứng dụng.

Xác định các kịch bản chính

- Các kịch bản chính có thể được định nghĩa là bất kỳ tình huống nào đáp ứng một hay nhiều tiêu chí sau:

  ✓ *Nó đại diện một vấn đề - Một khu vực không biết quan trọng hoặc một khu vực rủi ro quan trọng.*

  ✓ *Nó đề cập đế một use case quan trọng về mặt kiến trúc.*

  ✓ *Nó thể hiện sự giao nhau của các thuộc tính chất lượng với tính năng.*

  ✓ *Nó thể hiện sự cân bằng giữa các thuộc tính chất lượng.*

Xác định các kịch bản chính

- Ví dụ, việc xác thực người dùng có thể là các kịch bản chính bởi vì chúng là sự giao nhau của một thuộc tính chất lượng (bảo mật) với chức năng quan trọng (cách người dùng đăng nhập vào hệ thống).

- Một ví dụ khác là một kịch bản tập trung vào một công nghệ lạ và mới.

Các use case quan trọng

- Các use case quan trọng về mặt kiến trúc có ảnh hưởng đến nhiều khía cạnh của thiết kế. Những use case này thì đặc biệt quan trọng trong việc định hình sự thành công của ứng dụng.

- Chúng quan trọng đối với việc chấp nhận ứng dụng được triển khai, và chúng phải thực hiện đầy đủ trong thiết kế để có ích trong việc đánh giá kiến trúc.

Các use case quan trọng

Các use case quan trọng về mặt kiến trúc:

- Tiêu chí nghiệp vụ. *(Use case có mức độ sử dụng cao hay là đặc biệt quan trọng đối với người dùng hoặc những bên liên quan khác khi so sánh với các tính năng khác, hay nó tiềm ẩn rủi ro cao.)*

- Ảnh hưởng lớn. *(Use case giao nhau với cả các thuộc tính chức năng và chất lượng. Hay đại diện cho một mối quan tâm xuyên suốt có ảnh hưởng đầu cuối xuyên lớp và các tầng của ứng dụng. Một ví dụ, có thể là một hoạt động Create, Read, Update, Delete (CRUD) nhạy cảm với bảo mật.)*

Các use case quan trọng

- Những use case quan trọng được dùng để đánh giá thành công hay thất bại của các kiến trúc ứng viên.
- Nếu kiến trúc ứng viên giải quyết nhiều use case, hoặc các use case hiện có hiệu quả hơn. Điều này cho thấy kiến trúc ứng viên là một sự cải tiến so với kiến trúc cơ sở.
- Một use case tốt sẽ giao nhau giữa user view, system view, và business view của kiến trúc.
- Sử dụng các kịch bản và use case để kiểm tra thiết kế và xác định bất cứ nơi nào có thể có vấn đề.

Các use case quan trọng

Cần xem xét khi nghỉ đến các use case và kịch bản:

- Đầu dự án, giảm rủi ro bằng cách tạo ra một kiến trúc ứng viên hỗ trợ về mặt kiến trúc những kịch bản đầu cuối quan trọng, thục hiện trên tất cả các lớp của kiến trúc.
- Sử dụng mô hình kiến trúc như một hướng dẫn, tạo ra các thay đổi cho kiến trúc, thiết kế, và mã hóa để đáp ứng các kịch bản, yêu cầu chức năng, yêu cầu công nghệ, các thuộc tính chất lượng và các ràng buộc.
- Việc tạo mô hình kiến trúc dựa trên cái đã biết ở lúc đó, xác định một danh sách câu hỏi phải được giải quyết trong các việc tiếp theo và lặp lại.
- Sau khi thay đổi kiến trúc và thiết kế, hãy cân nhắc tạo ra một use case phản ánh và thực hiện thay đổi này

Tổng quan ứng dụng

Việc tạo một tổng quan cho ứng dụng sẽ cho thấy nó như thế nào khi hoàn chỉnh, giúp làm cho kiến trúc trở nên hữu hình hơn, kết nối nó đến các ràng buộc và các quyết định thế giới thực.

Tổng quan ứng dụng

Các hoạt động gồm:

- Xác định loại ứng dụng. *(Ứng dụng di động, khách hàng phong phú, ứng dụng Internet phong phú, một dịch vụ, một ứng dụng web, hay sự kết hợp của các loại)*
- Xác định các ràng buộc phát triển. *(Phải liệt kê các chính sách và thủ tục, cùng với cơ sở hạ tầng, mục tiêu cố định hay linh hoạt, nêu những hạn chế tồn tại)*
- Xác định kiểu thiết kế kiến trúc quan trọng.
- Xác định các công nghệ liên quan. *(Sự lựa chọn công nghệ cũng sẽ được điều chỉnh bởi các chính sách tổ chức, những hạn chế cơ sở hạ tầng, các khả năng nguồn tài nguyên và v.v.)*

Các công nghệ liên quan

Một số công nghệ trên nền tảng Microsoft:

- Các ứng dụng di động. *(.NET Compact Framework, ASP.NET for Mobile, và Silverlight for Mobile để phát triển các ứng dụng cho các thiết bị di động)*
- Các ứng dụng client phong phú. (Windows Presentation Foundation (WPF), Windows Forms, và XAML Browser Application (XBAP) để phát triển các ứng dụng với các giao diện người dùng phong phú chúng được phát triển và chạy trên client)

Các công nghệ liên quan

Một số công nghệ trên nền tảng Microsoft:

- Các ứng dụng Client Internet phong phú. *(Có thể sử dụng trình duyệt của Microsoft Silverlight, hay Siverlight được kết hợp với AJAX, để phát triển những trải nghiệm giao diện người dùng phong phú với một trình duyệt Web)*

- Các ứng dụng Web. *(Sử dụng ASP.NET Web Forms, AJAX, Silverlight controls, Service Applications. Sử dụng Windows Communication Foundation (WCF) và các dịch vụ ASP.NET Web (ASMX) để tạo các dịch vụ để trình bày tính năng cho các hệ thống bên ngoài và người sử dụng dịch vụ.)*

Các vấn đề chính

- Xác định các vấn đề trong kiến trúc ứng dụng để hiểu những lĩnh vực mà các sai xót thường xảy ra nhất. Những vấn đề tìm ẩn gồm việc xuất hiện của những công nghệ mới, và nhiều yêu cầu kinh doanh quan trọng.

- **Ví dụ:**

  "Tôi có thể chuyển đổi từ dịch vụ của bên thứ ba này sang dịch vụ của bên thứ ba khác không?,"

  "Tôi có thể bổ sung thêm hỗ trợ cho loại client mới không?,"

  "Tôi có thể nhanh chóng thay đổi các qui định kinh doanh liên quan đến thanh toán không?," và

  "Tôi có thể di chuyển sang một công nghệ mới X có được không?"

Các thuộc tính chất lượng

- Các thuộc tính chất lượng là những đặc trưng tổng thể của kiến trúc ảnh hướng đến hành vi thời gian chạy, thiết kế hệ thống, và kinh nghiệm người dùng.
- Mức ứng dụng có sự kết hợp các thuộc tính chất lượng chẳng hạng như: khả năng sử dụng, hiệu suất, độ tin cậy và bảo mật cho thấy sự thành công của thiết kế và chất lượng chung của ứng dụng phần mềm.
- Khi thiết kế các ứng dụng để đáp ứng bất kỳ tiêu chí chất lượng nào, nó cần phải xem xét tác động đến các yêu cầu khác; bạn phải phân tích sự cân bằng giữa nhiều thuộc tính chất lượng.

Các thuộc tính chất lượng

- Tầm quan trọng hay ưu tiên của mỗi thuộc tính chất lượng khác từ hệ thống đến hệ thống;
- Ví dụ, trong hệ thống kinh doanh (LOB), hiệu suất, khả năng mở rộng, bảo mật, và khả năng sử dụng sẽ quan trọng hơn khả năng tương tác. Khả năng tương tác có thể sẽ quan trọng hơn nhiều trong ứng khác hơn trong ứng dụng LOB.
- Các thuộc tính chất lượng biểu diễn lĩnh vực quan tâm, có khả năng ảnh hưởng đến toàn bộ ứng dụng xuyên qua các lớp và các tầng.
- Một số thuộc tính có liên quan đến thiết kế tổng thể, trong khi một số thuộc tính khác thì riêng biệt cho thời gian chạy, hay các vấn đề người dùng quan tâm

Các thuộc tính chất lượng

Một số thuộc tính:

- Các chất lượng hệ thống. *(Các chất lượng tổng quát của hệ thống khi được xem xét như một toàn thể; như khả năng hỗ trợ và khả năng kiểm tra)*
- Các chất lượng thời gian chạy. *(Những chất lượng của hệ thống được thể hiện trực tiếp trong thời gian chạy; như khả năng tiện lợi, khả năng tương tác, khả năng quản lý, tính hiệu quả, độ tin cậy, khả năng mở rộng và bảo mật)*
- Các chất lượng thiết kế. *(Những chất lượng phản ảnh thiết kế của hệ thống; như tính toàn vẹn khái niệm, tính linh hoạt, khả năng bảo trì, và khả năng tái sử dụng)*
- Các chất lượng người dùng. *(Khả năng sử dụng hệ thống)*

Những mối quan tâm xuyên suốt

Các mối quan tâm xuyên suốt là những đặc điểm của thiết kế có thể áp dụng trên tất cả các lớp, các thành phần, và các tầng.

Đây cũng là các lĩnh vực mà trong đó các lỗi thiết kế ảnh hưởng nhiều thường được tạo ra nhiều nhất.

Những mối quan tâm xuyên suốt

Các ví dụ về các mối quan tâm xuyên suốt:

- Xác thực và ủy quyền. *(Cách bạn chọn các chiến lược xác thực và ủy quyền thích hợp, nhận dạng luồng trên các lớp và các tầng, và những nhận dạng người dùng lưu trữ)*
- Vùng nhớ đệm (Cache). *(Cách bạn chọn công nghệ bộ nhớ đệm thích hợp, xác định dữ liệu nào lưu trên cache, nơi lưu dữ liệu cache, và chính sách hết hạn hợp lý)*
- Giao tiếp. *(Cách bạn chọn các giao thức thích hợp cho việc giao tiếp xuyên qua các lớp và các tầng, thiết kế khớp nối lỏng lẻo xuyên qua các lớp, thực hiện giao tiếp không đồng bộ, và truyền dữ liệu nhạy cảm)*
- Quản lý ngoại lệ. *(Cách bạn xử lý và đăng nhập các ngoại lệ, và cung cấp thông báo khi được yêu cầu)*

Những mối quan tâm xuyên suốt

Các ví dụ về các mối quan tâm xuyên suốt:

- Quản lý cấu hình. *(Cách bạn xác định thông tin nào phải được cấu hình, ở đâu và làm thế nào để lưu trữ thông tin cấu hình, làm thế nào để bảo vệ thông tin cấu hình nhạy cảm, cách xử lý thông tin cấu hình trong một farm hay cluster)*
- Đăng nhập và thiết bị. *(Cách bạn xác định thông tin để đăng nhập, làm thế nào để xây dựng cấu hình đang nhập, và xác định mức độ của thiết bị được yêu cầu)*
- Xác nhận. *(Làm thế nào xác định nơi nào và làm cách nào để thực hiện xác nhận; các kỹ thuật chọn để xác nhận về độ dài, phạm vi, định dạng, và loại; cách chọn hạn chế và từ chối các giá trị đầu vào không hợp lý; làm thế nào cải thiện đầu vào độc hại và nguy hiểm tiềm ẩn; và cách có thể định nghĩa và tái sử dụng việc xác định tính hợp lý xuyến suốt các lớp và các tầng ứng dụng)*

Thiết kế để giảm tải vấn đề

Các câu hỏi cần xem xét

- Kiểm tra và đăng nhập. Ai đã làm cái gì và khi nào? Ứng dụng có hoạt động bình thường không? *(Kiểm tra đề cập đến cách mà ứng dụng của bạn ghi lại những sự kiện có liên quan bảo mật. Đăng nhập đề cập đến cách mà ứng dụng phổ biến thông tin về các hoạt động.)*
- Xác thực. Bạn là ai? *(Xác thực là quá trình xử lý khi một thực thể thành lập rõ ràng xác định thực thể khác, thông thường thì xác nhận như tên người dùng và mật khẩu.)*
- Ủy quyền. Các gì bạn có thể làm? *(Ủy quyền ám chỉ làm thế nào ứng dung kiểm soát truy cập đến các nguồn tài nguyên và hoạt động.)*

Thiết kế để giảm tải vấn đề

Các câu hỏi cần xem xét

- Quản lý cấu hình. Bối cảnh nào mà ứng dụng của bạn chạy bên dưới? Cơ sở dữ liệu nào được kết nối đến? Ứng dụng của bạn được bảo vệ như thế nào? Các hệ thống này được bảo vệ như thế nào? *(Quản lý cấu hình ám chỉ làm thế nào mà ứng dụng xử lý các hoạt động và vấn đề đó.)*

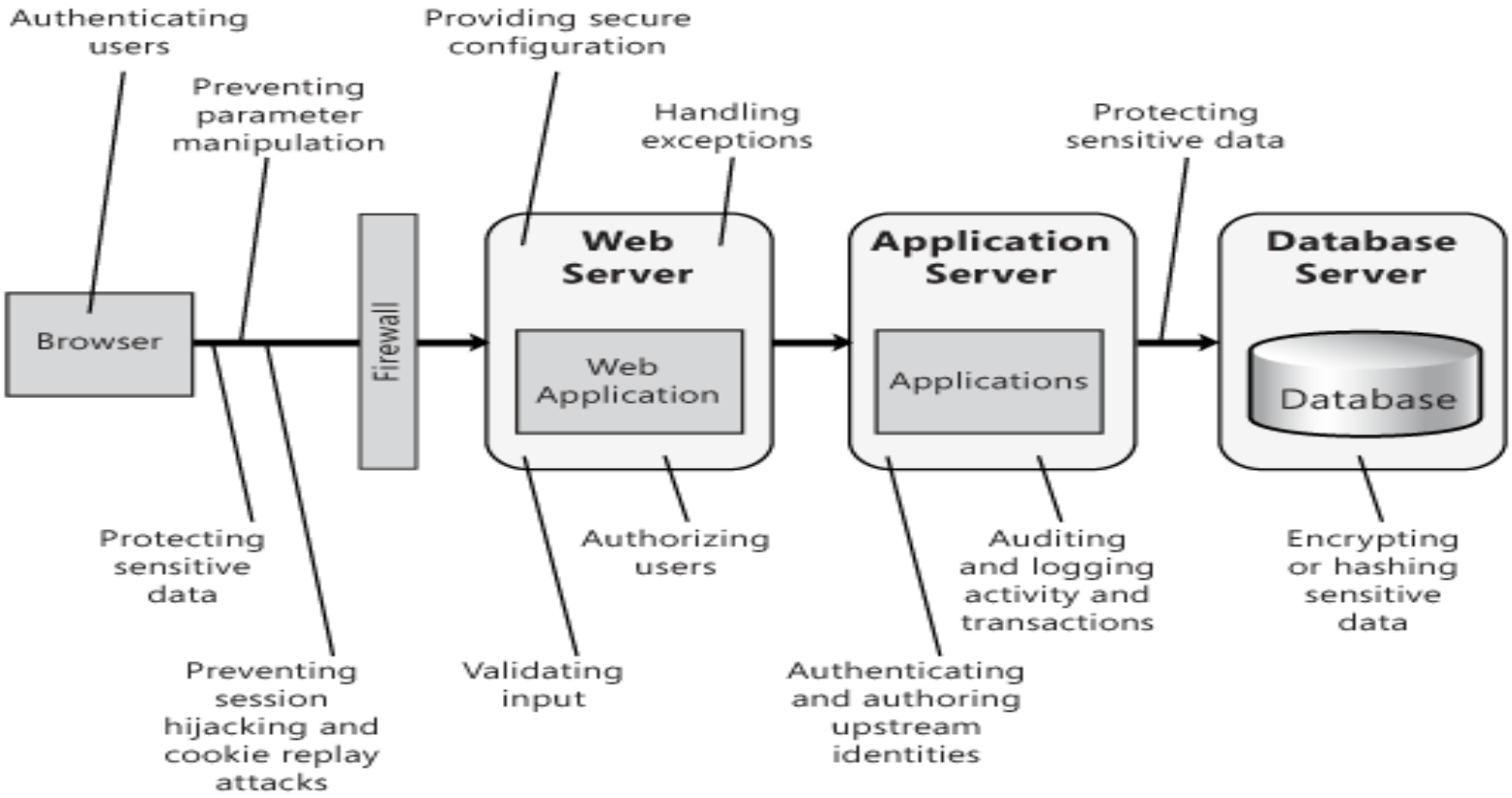
- Mật mã (Crytography). Bạn đang xử lý những bí mật như thế nào (bảo mật)? Làm thế nào mà bạn kiểm chức mạo danh dữ liệu hay các thư viện (toàn vẹn)? Làm thế nào gieo các giá trị ngẫu nhiên nó phải là mật mã mạnh? *(Mật mã ám chỉ cách mà ứng dụng thực thi bảo mật và toàn vẹn.)*

Thiết kế để giảm tải vấn đề

Các câu hỏi cần xem xét

- Nhập và xác nhận dữ liệu. Làm cách nào biết được đầu vào ứng dụng nhận được là hợp lệ và an toàn? *(Có ràng buộc đầu vào thông qua các điểm vào và mã hóa đầu ra thông qua các điểm ra.)* Có thể tin cậy các nguồn dữ liệu như cơ sở dữ liệu và chia sẽ file? *(Xác nhận đầu vào am chỉ cách thức ứng dụng lọc, tẩy tế bào chết, hay từ chối dữ liệu vào trước xử lý bổ sung.)*
- Dữ liệu nhạy cảm. Ứng dụng của bạn xử lý dữ liệu nhạy cảm như thế nào? Nó có bảo vệ dữ liệu người dùng và ứng dụng bí mật không? *(Dữ liệu nhạy cảm ám chỉ cách thức ứng dụng bạn xử lý bất kỳ dữ liệu nào cần được bảo vệ trong bộ nhớ, qua mạng, hay ở những nơi lưu trữ liên tục.)*
- Quản lý phiên. Ứng dụng của bạn xử lý và bảo vệ những phiên người dùng như thế nào? *(Một phiên ám chỉ tập tương tác có liên quan*

Thiết kế để giảm tải vấn đề



Bạn có thể sử dụng những câu hỏi và câu trả lời này để đưa ra những quyết định thiết kế bảo mật quan trọng cho ứng dụng, và tài liệu này là một phần của kiến trúc. Ví dụ, hình 3 cho thấy các vấn đề bảo mật được xác định trong một kiến trúc ứng dụng Web điển hình

Các kiến trúc ứng viên ban đầu

- Một kiến trúc ứng viên gồm *loại ứng dụng*, *kiến trúc phát triển*, *kiểu kiến trúc*, *các lựa chọn công nghệ*, *các thuộc tính chất lượng* và *mối quan tâm xuyên suốt*.

- Khi phát triển thiết kế hãy đảm bảo rằng ở mội giai đoạn bạn hiểu những rủi ro chính và thích ứng của thiết kế để giảm rủi ro, tối ưu hóa khả năng và hiệu quả giao tiếp thông tin thiết kế, xây dựng kiến một cách linh hoạt và tái cấu trúc trong trí nhớ.

- Kiến trúc có thể được sửa đổi qua một số bước lặp.

Các kiến trúc ứng viên ban đầu

Các câu hỏi kiểm tra kiến trúc ứng viên mới:

- Kiến trúc có thành công hay không mà không có bất kỳ rủi ro nào?
- Kiến trúc có giảm được nhiều rủi ro đã biết hơn những lần lặp trước hay không?
- Kiến trúc có đáp ứng được những yêu cầu bổ sung hay không?
- Kiến trúc có cho phép các use case quan trọng kiến trúc hay không?
- Kiến trúc có đề cập tới các mối quan tâm thuộc tính chất lượng hay không?
- Kiến trúc có giải quyết những mối quan tâm xuyên suốt bổ sung hay không?

Tiếp theo làm cái gì?

Sau khi hoàn thành hoạt động mô hình hóa kiến trúc, có thể bắt đầu tinh chỉnh thiết kế, kiểm tra kế hoạch, và trao đổi thiết kế. Nhớ một số hướng dẫn sau:

- Chuẩn bị một tài liệu bao gồm chi tiết các mục tiêu, loại ứng dụng, cấu trúc tài liệu, những kịch bản chính và các yêu cầu, công nghệ, các thuộc tính chất lượng và những thử nghiệm.
- Nên dự đoán trước tình huống xấu liên quan đến các rủi ro kiến trúc và giải pháp phù hợp giải quyết.
- Trao đổi thông tin bạn nắm bắt đến những thành viên nhóm liên quan và các bên liên quan khác *(như nhóm phát triển ứng dụng, nhóm kiểm thử, và những nhà quản trị mạng và hệ thống)*

Xem lại kiến trúc

- Xem lại kiến trúc là công việc rất quan trọng để giảm chi phí sai xót và tìm ra và sửa chữa những vấn đề kiến trúc càng sớm càng tốt.
- Xem lại kiến trúc là cách hiệu quả, tiết kiệm chi phí để giảm chi phí dự án và giảm cơ hội thất bại dự án.
- Mục tiêu chính của việc xem lại kiến trúc là xác định tính khả thi của kiến trúc ứng viên và cơ sở, và xác minh rằng kiến trúc liên kết đúng với những yêu cầu chức năng và các thuộc tính chất lượng với giải pháp kỹ thuật đã đề xuất.
- Ngoài ra, nó giúp xác định các vấn đề và nhận diện các khu vực để cải thiện.

Những đánh giá dựa trên kịch bản

- Đánh giá dựa trên kịch bản là một phương pháp tốt để xem lại một thiết kế kiến trúc.
- Đánh giá dựa trên kịch bản, tập trung những kịch bản quan trọng nhất từ quan điểm nghiệp vụ, và có tác động lớn nhất đến kiến trúc.

Những đánh giá dựa trên kịch bản

Một số phương pháp đánh giá phổ biến:

- Software Architecture Analysis Method (SAAM). *SAAM được thiết ban đầu cho đánh giá khả năng điều chỉnh, nhưng sau đó được mở rộng để xem lại kiến trúc có liên quan đến những yêu cầu thuộc tính chất lượng như tính dễ thay đổi, tính di động, khả năng mở rộng, khả năng kết hợp, và bao phủ chức năng*
- Architecture Tradeoff Analysis Method (ATAM). *ATAM là phiên bản được cải tiến và tinh chỉnh của SAAM nhằm giúp xem lại những quyết định kiến trúc đối với các yêu cầu thuộc tính chất lượng, và mức độ đáp ứng các mục tiêu chất lượng cụ thể*
- Active Design Review (ADR). *ADR là phù hợp nhất cho những kiến trúc không đầy đủ hoặc đang tiến hành. Sự khác biệt chính là việc xem xét tập trung nhiều trên tập các vấn đề hay từng phần riêng lẻ của kiến trúc tại một thời điểm, chứ không phải thực hiện một xem xét tổng quan.*

Những đánh giá dựa trên kịch bản

Một số phương pháp đánh giá phổ biến:

- Active Reviews of Intermediate Designs (ARID). *ARID kết hợp khía cạnh ADR trong việc xem xét kiến trúc đang thực hiện với sự tập trung trên tập các vấn đề, và hướng tiếp cận ATAM và SAAM được tập trung vào các thuộc tính chất lượng.*
- Cost Benefit Analysis Method (CBAM). *CBAM được tập trung trên việc phân tích các chi phí, các lợi ích, và những liên can về lịch trình của các quyết định kiến trúc.*
- Architecture Level Modifiability Analysis (ALMA). *ALMA đánh giá khả năng thay đổi kiến trúc đối với các hệ thống thông tin kinh doanh (BIS).*
- Family Architecture Assessment Method (FAAM). *FAAM đánh giá các kiến trúc hệ thống thông tin đối với khả năng tương tác và mở rộng.*

Trình bày và trao đổi thiết kế kiến trúc

- Trao đổi thiết kế quan trọng cũng như xem lại kiến trúc nhằm đảm bảo nó được thực hiện chính xác.

- Trao đổi thiết kế kiến trúc đến tất cả các bên có liên quan gồm nhóm phát triển, các nhà quản trị và người khai thác, chủ sở hữu kinh doanh, và các bên quan tâm khác.

Trình bày và trao đổi thiết kế kiến trúc

Một số phương pháp nổi tiếng mô tả kiến trúc:

- 4+1. Hướng tiếp cận sử dụng 5 quan điểm cho một kiến trúc hoàn chỉnh.
  - ✓ Quan điểm lôgic (như mô hình đối tượng).
  - ✓ Quan điểm xử lý (như các khía cạnh đồng thời và đồng bộ hóa)
  - ✓ Quan điểm vật lý (sơ đồ các lớp và chức năng phần mềm trên cơ sở hạ tầng cứng phân tán), và
  - ✓ Quan điểm phát triển
  - ✓ Quan điểm các kịch bản và use case cho phần mềm.

Trình bày và trao đổi thiết kế kiến trúc

Một số phương pháp nổi tiếng mô tả kiến trúc:

- UML. Hướng tiếp cận thể hiện 3 quan điểm cho một mô hình hệ thống.
  - ✓ Quan điểm các yêu cầu chức năng *(chức năng của hệ thống từ quan điểm của người dùng gồm các use case).*
  - ✓ Quan điểm cấu trúc tĩnh *(các đối tượng, thuộc tính, mối quan hệ, và các hoạt động gồm các lược đồ lớp)*
  - ✓ Quan điểm hành vi động *(sự hợp tác giữa các đối tượng và thay đổi trạng thái bên trong của đối tượng, gồm trình tự, hoạt động và các lược đồ trạng thái)*


Q&A