ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

NGUYỄN MINH HẰNG

PHƯƠNG PHÁP SINH TỰ ĐỘNG BẢN MẪU GIAO DIỆN NGƯỜI DÙNG TỪ ĐẶC TẢ YÊU CẦU CHỨC NĂNG

Ngành: Công nghệ thông tin
Chuyên ngành: Kỹ thuật phần mềm
Mã số: 8480103.01

LUẬN VĂN THẠC SĨ CÔNG NGHỆ THÔNG TIN

CÁN BỘ HƯỚNG DẪN: TS. ĐẶNG ĐỨC HẠNH

Hà Nội – 2020

TÓM TẮT

Tóm tắt: Giao diện người dùng (User Interface) rất quan trọng trong thời đại ứng dụng web và di động ngày nay. Do đó, trong các giai đoạn đầu của Vòng đời phát triển phần mềm (Software Development Life Cycle), việc phát triển giao diện người dùng chính xác là vô cùng cần thiết. Để đạt được điều này, Nhóm Quản lý Đối tượng (Object Management Group - OMG) đã giới thiệu tiêu chuẩn Ngôn ngữ mô hình hóa luồng tương tác (Interaction Flow Modeling Language - IFML) vào năm 2013. IFML cung cấp mô hình giao diện người dùng cho các ứng dụng đa dạng như thiết bị di động, web và máy tính. Mặc dù IFML dựa trên nguyên tắc của kỹ nghệ hướng mô hình (Model Driven Engineering - MDE), sự phát triển của các mô hình giao diện người dùng từ các yêu cầu ban đầu là công việc phức tạp và tốn thời gian. Đặc biệt, nó đòi hỏi kiến thức chuyên môn về miền và hiểu biết một số khái niệm của IFML như view container, view component, event, ... Do đó, cách tiếp cận để tự động hóa việc phát triển bản mẫu giao diện người dùng từ các đặc tả yêu cầu chức năng ban đầu là nhu cầu vô cùng cần thiết. Luận văn này giới thiệu một phương pháp để tự động tạo các mô hình IFML từ các đặc tả yêu cầu chức năng dạng văn bản bằng cách sử dụng các tính năng của xử lý ngôn ngữ tự nhiên (Natural Language Processing - NLP). Sau đó, một tập các luật được áp dụng để trích xuất các phần tử quan trọng của IFML như view container, view component, event, action ... từ văn bản đầu vào. Từ đó, kết hợp các đặc tả yêu cầu văn bản cho sơ đồ lớp và sơ đồ ca sử dụng để sinh ra json file biểu diễn mô hình IFML. Từ mô hình IFML có thể sinh tự động bản mẫu giao diện người dùng mong muốn. Cuối cùng, các ví dụ đơn giản cũng được trình bày trong luận văn nhằm đánh giá tính khả thi và tính hiệu quả của phương pháp.

Từ khóa: UI, IFML, NLP, MDE

MỤC LỤC

LỜI CẢM ƠN

Trong suốt quá trình làm nghiên cứu, bên cạnh những nỗ lực của bản thân, tôi còn nhận được sự hỗ trợ rất lớn từ phía nhà trường và giảng viên hướng dẫn, cũng như bạn bè trong nhóm nghiên cứu.

Trước hết, tôi muốn gửi lời cảm ơn chân thành đến giảng viên hướng dẫn, Tiến Sĩ Đặng Đức Hạnh - hiện đang công tác tại bộ môn Công Nghệ Phần Mềm, người đã tận tâm hướng dẫn và chỉ dạy tôi để hoàn thành luận văn này. Tôi cũng xin cảm ơn sự hỗ trợ của đề tài nghiên cứu khoa học mã số QG.20.54 của Đại học Quốc gia Hà Nội. Tôi cũng xin gửi lời cảm ơn về phía nhà trường đã hỗ trợ tối đa về điều kiện vật chất và giúp đỡ tôi trong quá trình nghiên cứu và thực hiện luận văn.

Cuối cùng, tôi muốn gửi lời cảm ơn đến các bạn trong lớp, những người đã động viên, giúp đỡ tôi trong quá trình thực hiện luận văn.

Hà Nội, tháng 12 năm 2020

Học viên thực hiện

Nguyễn Minh Hằng

LỜI CAM ĐOAN

Tôi là Nguyễn Minh Hằng, học viên cao học khóa K24 - CNPM của Trường Đại học Công nghệ - Đại học Quốc gia Hà Nội. Tôi xin cam đoan đây là công trình nghiên cứu của tôi dưới sự giúp đỡ rất lớn của Giảng viên hướng dẫn là Tiến sĩ Đặng Đức Hạnh và các bạn trong nhóm Nghiên cứu. Những nội dung nghiên cứu và kết quả trong đề tài này là hoàn toàn trung thực. Các trích dẫn từ tài liệu bên ngoài tôi đều liệt kê rõ ràng ở cuối của luận văn.

Hà Nội, tháng 12 năm 2020

Học viên thực hiện

Nguyễn Minh Hằng

DANH MỤC HÌNH ẢNH

DANH MỤC BẢNG BIỂU

DANH MỤC CÁC KÝ HIỆU VÀ CHỮ VIẾT TẮT

Ký hiệu	Nội dung
UI	User Interface – Giao diện người dùng
SDLC	Software Development Life Cycle – Vòng đời phát triển phần mềm
MDE	Model Driven Engineering – Kỹ nghệ hướng mô hình
IFML	Interaction Flow Modeling Language – Ngôn ngữ mô hình hóa luồng tương tác

CHƯƠNG 1. GIỚI THIỆU

1.1. Đặt vấn đề

Giao diện người dùng (User Interface - UI) là một phần rất quan trọng trong các ứng dụng web và di động ngày nay. Các ứng dụng này ngày càng có các giao diện đa dạng và phức tạp nên việc phát triển giao diện người dùng trong giai đoạn đầu của vòng đời phát triển phần mềm (Software Development Life Cycle - SDLC) vô cùng tốn kém công sức và thời gian. Người ta ước tính rằng việc phát triển một ứng dụng yêu cầu 48% code được áp dụng trên các giao diện người dùng và 50% thời gian phát triển được dành cho việc biểu diễn các giao diện người dùng [1]. Do đó, quá trình phát triển từ các yêu cầu đến thiết kế đã làm dấy lên những lo ngại liên quan đến việc tăng cường tiêu thụ nguồn lực và thời gian.

Kỹ nghệ hướng mô hình (Model Driven Engineering - MDE) có xu hướng đơn giản hóa kiến trúc của hệ thống bằng cách giới thiệu các mô hình cung cấp các mức độ trừu tượng khác nhau. Theo MDE, các lớp trừu tượng khác nhau sử dụng các mô hình khác nhau để xác định một hệ thống. Do đó, một phép biến đổi mô hình tự động được phát triển để lấy lớp trừu tượng thấp hơn từ lớp trừu tượng cao hơn [2]. Điều này dẫn đến việc tách biệt các phức tạp của lập trình khỏi nền tảng thực thi. Hơn nữa, năng suất được tăng lên và việc tái sử dụng thiết kế hệ thống cũng được đảm bảo và việc duy trì khả năng truy xuất nguồn gốc, tính đầy đủ và nhất quán cũng đạt được. Do những lợi ích của MDE đã nói ở trên, ngày nay nó thường được áp dụng để phát triển các giao diện người dùng.

Cho đến năm 2013, chưa có tiêu chuẩn MDE nào có sẵn để áp dụng các khái niệm về mô hình trừu tượng cho giao diện người dùng ngoại trừ ngôn ngữ mô hình hóa web WebML[3] và nó chỉ dành cho các ứng dụng web. Để giải quyết vấn đề này, ngôn ngữ mô hình hóa luồng tương tác (IFML) đã được OMG giới thiệu vào tháng 3 năm 2013 [4]. Đặc biệt, IFML dựa trên các khái niệm WebML và cung cấp mô hình hóa các giao diện người dùng tinh vi cho các ứng dụng khác nhau như di động, web. Nó là một ngôn ngữ mô hình hóa độc lập nền tảng do đó được triển khai ở cấp Mô hình Độc lập Nền tảng (Platform Independent Model - PIM). Nó cho phép nắm bắt tương tác của người dùng, nội dung của các giao diện người dùng front-end và lập mô hình hành vi kiểm soát của các giao diện người dùng của hệ thống chủ thể [5].

Trong Vòng đời phát triển phần mềm (SDLC), phân tích yêu cầu là một giai đoạn quan trọng. Các yêu cầu được phân tích ban đầu đóng vai trò là đầu vào cho các giai đoạn SDLC tiếp theo. Thông thường, các yêu cầu được đưa ra từ người dùng bằng ngôn ngữ tự nhiên vì nó thúc đẩy quyền tự do ngôn luận. Tuy nhiên, nó cũng tạo ra nhiều vấn đề cho các bên liên quan khác nhau do tính chất dễ mắc lỗi và không rõ ràng vì các yêu cầu được

viết bằng ngôn ngữ tự nhiên thuần túy có thể được diễn giải theo nhiều cách khác nhau [6]. Để quản lý các vấn đề như vậy, các mô hình hoặc bản mẫu khác nhau được phát triển để thực hiện xác nhận ban đầu của các yêu cầu. IFML đang cho thấy các tính năng đầy hứa hẹn để phát triển các mô hình/bản mẫu giao diện người dùng ban đầu từ các yêu cầu.

Điều này dẫn đến việc thực hiện xác nhận các yêu cầu chính trong giai đoạn đầu. Hơn nữa, nó cũng giảm thời gian phát triển vì các mô hình IFML đã tạo có thể được sử dụng trong các giai đoạn SDLC tiếp theo bằng cách áp dụng các phép chuyển đổi mô hình. Mặc dù IFML dựa trên nguyên tắc MDE, việc phát triển các mô hình giao diện người dùng từ các yêu cầu ban đầu vẫn là công việc phức tạp và tốn thời gian. Đặc biệt, nó đòi hỏi kiến thức chuyên môn về miền để hiểu một số khái niệm IFML như mô hình miền, view container, event, ... để có thể mô hình hóa giao diện người dùng phù hợp. Do đó, có một yêu cầu mạnh mẽ về cách tiếp cận để tự động hóa việc phát triển các mô hình IFML từ các yêu cầu văn bản thuần túy ban đầu. Điều này có thể đạt được bằng cách sử dụng các tính năng của xử lý ngôn ngữ tự nhiên.

Luận văn này cung cấp một phương pháp sinh tự động bản mẫu giao diện người dùng từ đặc tả yêu cầu chức năng thông qua mô hình hóa luồng tương tác IFML.

1.2. Mục tiêu và phương pháp

Luận văn đưa ra mục tiêu xây dựng một phương pháp sinh tự động bản mẫu giao diện người dùng từ các đặc tả yêu cầu ban đầu dựa trên kỹ thuật mô hình hóa nhằm tách biệt các phức tạp của lập trình khỏi nền tảng thực thi, và tăng hiệu suất phát triển nhờ việc tái sử dụng thiết kế hệ thống.

Đầu tiên, luận văn áp dụng việc xử lý ngôn ngữ tự nhiên và các luật chuyển đổi để sinh tự động các thành phần của mô hình IFML từ đặc tả yêu cầu được viết bằng ngôn ngữ tự nhiên. Tiếp theo, kết hợp các thành phần IFML với các đặc tả yêu cầu chức năng mô hình ca sử dụng và mô hình lớp dạng văn bản để sinh ra json file biểu diễn mô hình IFML. Từ đó, sử dụng công cụ IFML Editor để sinh tự động bản mẫu giao diện người dùng từ mô hình IFML. Cuối cùng, các ví dụ đơn giản mô phỏng phương pháp sinh bản mẫu giao diện người dùng được trình bày.

1.3. Bố cục luận văn

- Chương 1: Giới thiệu, trình bày vấn đề nghiên cứu, mục tiêu và phương pháp sử dụng

- Chương 2: Kiến thức nền tảng, trình bày các cơ sở lý thuyết về mô hình IFML và thư viện xử lý ngôn ngữ tự nhiên OpenNPL được áp dụng để sinh bản mẫu giao diện.
- Chương 3: Trình bày phương pháp sinh tự động bản mẫu giao diện người dùng từ đặc tả yêu cầu chức năng.
- Chương 4: Đưa ra ví dụ minh họa để đánh giá tính khả thi của phương pháp.
- Chương 5: Tóm tắt những kết quả đạt được, các hạn chế còn tồn tại và hướng phát triển trong tương lai.

CHƯƠNG 2. KIẾN THỨC NỀN TẢNG

Để có thể áp dụng phương pháp sinh tự động bản mẫu giao diện người dùng từ đặc tả yêu cầu chức năng, luận văn cần tìm hiểu các kiến thức nền tảng về kỹ nghệ hướng mô hình MDE, mô hình hóa luồng tương tác IFML, các đặc tả yêu cầu chức năng và thư viện xử lý ngôn ngữ tự nhiên OpenNLP.

2.1. Giới thiệu

Chương này trình bày các cơ sở lý thuyết về các kỹ thuật, ngôn ngữ và công cụ hỗ trợ được sử dụng trong luận văn. Phần đầu chương, luận văn tìm hiểu về kỹ nghệ hướng mô hình MDE, trình bày các khái niệm và các cấp độ mô hình hóa của nó. Tiếp theo, luận văn trình bày lý thuyết về ngôn ngữ mô hình hóa luồng tương tác IFML bao gồm khái niệm và các thành phần chính của IFML. Sau đó, luận văn nêu ra các kiến thức nền tảng về các đặc tả yêu cầu chức năng như đặc tả yêu cầu bằng ngôn ngữ tự nhiên, đặc tả ca sử dụng và đặc tả sơ đồ lớp dạng văn bản. Phần cuối sẽ tìm hiểu về thư viện xử lý ngôn ngữ tự nhiên OpenNLP nhằm hỗ trợ quá trình xử lý đặc tả yêu cầu.

2.2. Kỹ nghệ hướng mô hình (MDE)

Mô hình có vai trò quan trọng trong việc hiểu và chia sẻ kiến thức về các phần mềm phức tạp. Kỹ nghệ hướng mô hình (Model-Driven Engineering MDE) được hình thành như một phương pháp để chuyển đổi mô hình trong công nghệ phần mềm.

2.2.1. Giới thiệu MDE

Kỹ nghệ hướng mô hình (MDE) có thể được định nghĩa là một phương pháp luận để áp dụng các lợi thế của mô hình hóa vào các hoạt động kỹ thuật phần mềm [7]. Mục đích của các mô hình có thể trải dài từ giao tiếp giữa mọi người tới khả năng thực thi của phần mềm được thiết kế: cách thức mà các mô hình được xác định và quản lý sẽ dựa trên nhu cầu thực tế mà nó sẽ giải quyết. Do các nhu cầu có thể khác nhau mà MDE giải quyết, vai trò của MDE trở thành xác định các phương pháp tiếp cận kỹ thuật đối với định nghĩa về mô hình, biến đổi và kết hợp chúng trong quy trình phát triển phần mềm.

Theo phương trình nổi tiếng về chương trình phần mềm của Niklaus Wirth:

Thuật toán + Cấu trúc dữ liệu = Chương trình [7]

Trong ngữ cảnh MDE mới, phương trình trở lên đơn giản hơn:

Mô hình + Phép chuyển đổi = Phần mềm [7]

Cả mô hình và các phép chuyển đổi cần được thể hiện trong một dạng ký hiệu hoặc ngôn ngữ nào đó, trong MDE, nó được gọi là ngôn ngữ mô hình hóa. Theo cách tương tự như trong phương trình Wirth, các thuật toán và cấu trúc dữ liệu được định nghĩa trong

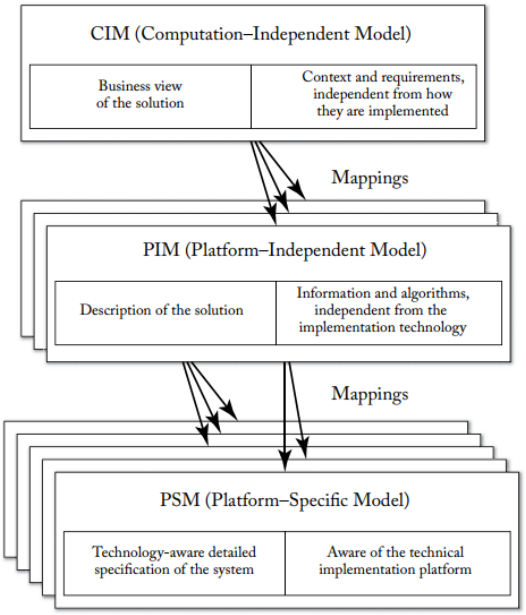
một số ngôn ngữ lập trình. Tuy nhiên, phương trình này không cho biết được loại mô hình nào (theo thứ tự nào, ở mức trừu tượng nào, v.v.) được xác định. Do đó cần có một bộ công cụ thích hợp để làm cho MDE khả thi trong thực tế. Đối với chương trình, cần có các IDE cho phép xác định các mô hình và các phép biến đổi cũng như các trình biên dịch hoặc trình thông dịch để thực thi chúng và tạo ra các tạo phẩm phần mềm cuối cùng.

MDE rất coi trọng tuyên bố "mọi thứ đều là một mô hình". Thực tế, trong bối cảnh này, người ta có thể nghĩ ngay đến tất cả các thành phần được mô tả ở trên là một thứ có mô hình hóa được. Cụ thể, có thể thấy các phép biến đổi là các mô hình hoạt động cụ thể dựa trên các mô hình. Định nghĩa của một ngôn ngữ mô hình hóa có thể được xem như là một mô hình: MDE gọi thủ tục này là siêu mô hình (memtamodel) (nghĩa là mô hình hóa một mô hình, hoặc mô hình hóa một ngôn ngữ mô hình hóa, hoặc mô hình hóa tất cả các mô hình có thể được biểu diễn bằng ngôn ngữ). Theo cách này, có thể định nghĩa là các mô hình cũng là các quy trình, công cụ phát triển và các chương trình kết quả. [14]

2.2.2. Các cấp độ mô hình hóa

Mức độ trừu tượng của các mô hình có thể thay đổi tùy thuộc vào mục tiêu của các mô hình. Có ba mức trừu tượng khác nhau được đưa ra cho các mô hình trong MDE [7], được chỉ ra trong Hình 2.1.

- **Mô hình độc lập tính toán (Computation-Independent Model-CIM**): cấp độ trừu tượng nhất. Đại diện cho bối cảnh, yêu cầu và mục đích của giải pháp mà không có bất kỳ ràng buộc nào đối với việc tính toán. Nó trình bày chính xác những gì giải pháp dự kiến sẽ làm, nhưng ẩn các thông số kỹ thuật công nghệ thông tin, để độc lập với việc hệ thống sẽ được thực hiện như thế nào.
- **Mô hình độc lập với nền tảng (Platform-Independent Mode - PIM):** Cấp độ mô tả hành vi và cấu trúc của ứng dụng, mà không phụ thuộc nền tảng triển khai. Lưu ý rằng PIM chỉ dành cho một phần của CIM mà được giải quyết bằng giải pháp dựa trên phần mềm và tinh chỉnh nó theo các yêu cầu đối với hệ thống phần mềm. PIM thể hiện một mức độ độc lập đủ để cho phép ánh xạ của nó tới một hoặc nhiều nền tảng triển khai cụ thể.
- **Mô hình dành riêng cho nền tảng (Platform-Specific Mode-PSM):** mô hình chứa tất cả thông tin bắt buộc liên quan đến hành vi và cấu trúc của một ứng dụng trên một nền tảng cụ thể mà các nhà phát triển có thể sử dụng để triển khai mã thực thi.



Hình 2.1. Ba mức độ trừu tượng của mô hình hóa trong MDA [7].

2.3. Mô hình hóa luồng tương tác (IFML)

Mục này tập trung tìm hiểu ngôn ngữ mô hình hóa luồng tương tác IFML bao gồm khái niệm, tính chất và vai trò của nó trong quá trình thiết kế giao diện. Đồng thời, cú pháp và ngữ nghĩa của các thành phần cấu trúc của IMFL cũng được trình bày.

2.3.1. Khái niệm

Thiết kế bậc giao diện của một hệ thống về cơ bản phụ thuộc vào các ca sử dụng chi tiết hoặc sơ đồ trình tự hệ thống, vì giao diện phải cho phép người dùng theo dõi tất cả các luồng ca sử dụng. Thiết kế sơ đồ lớp và hợp đồng cũng có thể rất hữu ích. Trong quá trình thiết kế giao diện, các yêu cầu phi chức năng đã được chú thích với các ca sử dụng phải được sửa đổi lại, vì chúng có thể chứa các chỉ dẫn về cách thiết kế giao diện ca sử dụng.

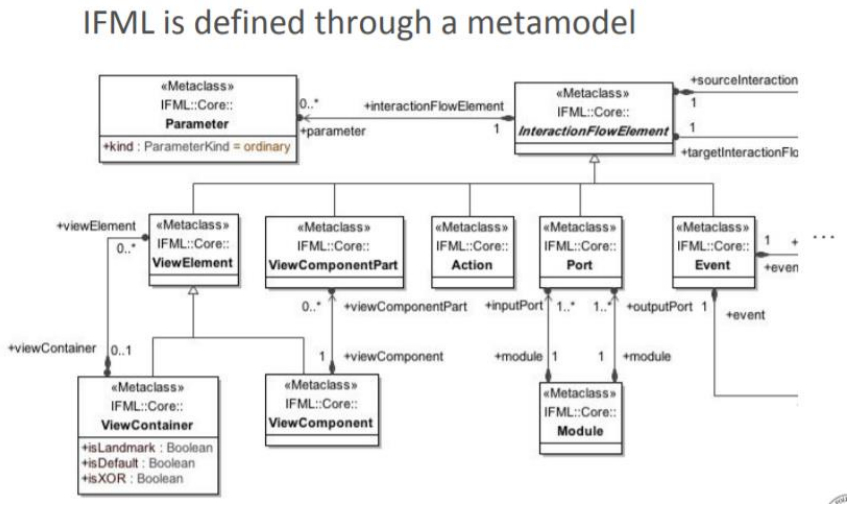
Ngôn ngữ mô hình hóa luồng tương tác (IFML) là một phần mở rộng của UML cho các giao diện người dùng chuyên sâu về dữ liệu của mô hình. Nó là một sự phát triển của WebML (Ceri, Fraternali, Bongio, Brambilla, Comai và Matera, 2003). IFML đang được chấp nhận như một tiêu chuẩn OMG.

IFML nhằm mục đích mô tả hành vi và cấu trúc của ứng dụng theo quan điểm của người dùng [15]. Tuy nhiên, mô tả hành vi và cấu trúc của ứng dụng trong ngữ cảnh logic nghiệp vụ và thành phần dữ liệu bị giới hạn ảnh hưởng trực tiếp đến trải nghiệm người dùng [15]. IFML hỗ trợ việc tích hợp với các mô hình khác để chỉ rõ các tính năng của ứng dụng một cách hoàn chỉnh.

IFML là một ngôn ngữ mô hình hóa độc lập nền tảng do đó được triển khai ở cấp độ mô hình độc lập nền tảng (PIM) để thể hiện các quyết định thiết kế tương tác độc lập với nền tảng triển khai. Nó cho phép nắm bắt tương tác của người dùng, nội dung của các giao diện người dùng front-end và mô hình hóa hành vi kiểm soát của các giao diện người dùng của hệ thống. [8]

2.3.2. Cú pháp và ngữ nghĩa của IFML

Ngôn ngữ mô hình luồng tương tác bao gồm một số các thành phần để mô hình hóa các yêu cầu giao diện người dùng. Hình 2.2 bên dưới trình bày một metamodel của IFML, nó thể hiện các thành phần cấu trúc của mô hình IFML và mối quan hệ giữa các thành phần này.



Hình 2.2. Metamodel của mô hình IFML

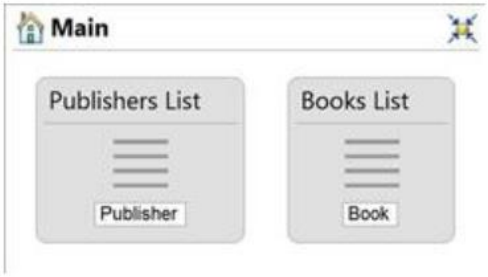
Các phần tử IFML quan trọng được tập trung tìm hiểu và nghiên cứu trong luận văn là: View Container, View Component, Event và Action.

2.3.2.1. Thành phần View Container

View Container là một trong những phần tử dạng View Elements của mô hình IFML, phần có thể nhìn thấy ở cấp giao diện người dùng. Một số thuộc tính của View Containner là:

- Mô hình IFML bao gồm một hoặc nhiều View Container, được sử dụng để thể hiện các web page trong trường hợp ứng dụng web hay window trong ứng dụng máy tính để bàn.
- View Container biểu diễn nội dung của giao diện. Giống như các trang html, nó có thể chứa một hoặc nhiều thành phần View Component như form, detail hoặc list.

Ví dụ về Book Store Online được đưa ra trong chương này để minh họa các thành phần của mô hình IFML. Hình 2.3 biểu diễn một View Container Main của giao diện ứng dụng Book Store chứa 2 phần tử View Component là list of publishers and list of books. Và Hình 2.4 thể hiện giao diện tương ứng được sinh ra từ mô hình của View Container Main.



Hình 2.3. Thành phần View Container Main.

Publishers List

name	city
Andrews and McMeel	Kansas City
Bantam	New York
Boxtree	London
Pocket Books	New York
Random House	New York

Books List

isbn	title	authorsName	price	pageCount	quantityInStock
0671746723	Dirk Getly's Holistic Detective Agency	Douglas N. Adams	6.99	306	12
0553293370	Foundation and Empire	Isaac Asimov	5.99	282	3
0553286587	Rama II	Arthur C. Clarke and Gentry Lee	6.99	466	2
0752208497	Shave the Whales	Scott Adams	6.99	128	7
055356871X	The Hammer of God	Arthur C. Clarke	6.99	240	1
0671742515	The Long Dark Tea-Time of the Soul	Douglas N. Adams	6.99	307	21
0836218663	The Revenge of the Baby-Sat	Bill Waterson	8.95	127	4
0517149257	The Ultimate Hitchhikers Guide	Douglas N. Adams	129	813	6

Hình 2.4. Giao diện được sinh ra từ View Container Main.

2.3.2.2. Thành phần View Component

View Component là thành phần dạng View Elements cơ bản nhất trong đặc tả IFML. Một View Component không bao giờ có thể tồn tại bên ngoài View Container.

View Component bao gồm nhiều loại khác nhau như:

- Details: Hiển thị thông tin về một đối tượng duy nhất.
- Multiple Details: Hiển thị thông tin về tập hợp các thực thể (instances) của cùng một lớp.

- Simple List: Hiển thị nhiều thực thể của một lớp dưới dạng danh sách.
- List: Một cải tiến của Simple List cho phép cuộn và sắp xếp động.
- Checkable list: Danh sách cho phép nhiều lựa chọn.
- Hierarchy: Hiển thị nhiều thực thể của các lớp khác nhau được tổ chức theo hệ thống phân cấp, ví dụ như chi tiết tổng thể hoặc toàn bộ.
- Recursive hierarchy: Hiển thị nhiều thực thể của một lớp đơn lẻ được tổ chức phân cấp.
- Scroller: Cho phép thực hiện các thao tác cuộn trên một chuỗi các đối tượng.
- Form: Cho phép nhập dữ liệu dựa trên biểu mẫu.

IFML vẫn còn nhiều hơn các thành phần khác nữa nhưng chương này chỉ tập trung trình bày các thành phần cơ bản nhất. Mỗi loại thành phần View Component sẽ có một định nghĩa đồ họa tương ứng thể hiện nó.

Details: Trình bày thông tin về một đối tượng duy nhất của một lớp nhất định. Nó được xác định bởi các thuộc tính sau:

- Tên: được chọn người thiết kế mô hình
- Thực thể: một lớp trong sơ đồ lớp
- Thuộc tính hiển thị: là danh sách các thuộc tính của lớp sẽ được hiển thị

Thành phần Details phép định nghĩa một hoặc nhiều biểu thức điều kiện hoạt động như bộ lọc trên tập hợp các thực thể được hiển thị. Có ba loại biểu thức điều kiện:

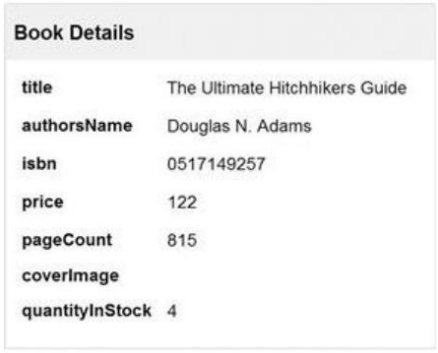
- Điều kiện khóa (Key condition): Mỗi đối tượng trong mô hình dữ liệu được xác định bởi một định danh của đối tượng OID (Object IDentifier) - một thuộc tính là duy nhất, bắt buộc và bất biến. OID là mã được tạo bên trong tương ứng với khóa chính của một đối tượng. Người dùng không thể biết hoặc cập nhật OID. Điều kiện khóa cho phép người ta chọn một hoặc nhiều thực thể từ một lớp nhất định dựa trên giá trị của thuộc tính OID của nó.
- Điều kiện thuộc tính (Attribute condition): Điều kiện cho phép lựa chọn một hoặc nhiều thực thể dựa trên các giá trị của thuộc tính của chúng. Các phép toán như lớn hơn, bằng hoặc bao gồm có thể được sử dụng để so sánh giá trị của một thuộc tính với một tham số nhất định.
- Điều kiện vai trò quan hệ (Relationship role condition) : Điều kiện cho phép lựa chọn một hoặc nhiều thể hiện dựa trên các liên kết của chúng.

Hình 2.5 trình bày một thành phần Details nhằm hiển thị dữ liệu về một bản sao của sách cho một mã sách tiêu chuẩn quốc tế ISBN (International Standard Book Number)

nhất định. Biểu thức điều kiện isbn = 0517149257 xác định thực thể nào được hiển thị. Nếu một thực thể như vậy không tồn tại, thì không có thông tin nào được hiển thị bởi thành phần đó.

Hình 2.5. Thành phần Details với điều kiện thuộc tính đơn giản.

Hình 2.6 bên dưới biểu diễn giao diện tương ứng của thành phần Details mô tả trong Hình 2.5.

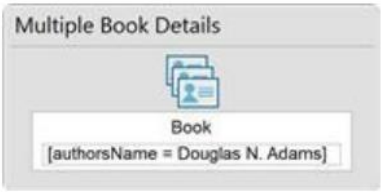


Hình 2.6. Giao diện tương ứng cho thành phần Details với các thuộc tính của lớp Book.

Multiple Details: trình bày một nhóm các thực thể của một lớp cùng một lúc. Bên cạnh ba thuộc tính đã nêu trên của thành phần Details, Multiple Details còn bao gồm thêm các thuộc tính tùy chọn mới:

- Sắp xếp thuộc tính (Sort attributes): Xác định các thuộc tính được sử dụng để sắp xếp các thực thể.
- Kết quả tối đa (Max results): Chỉ định số lượng thực thể tối đa được truy xuất. Theo mặc định, tất cả các thực thể đều được truy xuất.
- Phân biệt (Distinct): Chỉ định rằng các phần tử trùng lặp sẽ không được hiển thị nhiều lần

Multipel Details cũng cho phép định nghĩa và sử dụng các biểu thức điều kiện để chọn các phần tử sẽ được hiển thị. Hình 2.7 trình bày một ví dụ về MultiPle Details.



Hình 2.7. Thành phần Multiple Details.

Hình 2.8 thể hiện giao diện tương tứng của thành phần Multiple Details trong Hình 2.7. Có 3 cuốn sách đáp ứng được biểu thức điều kiện của Hình 2.7 và được sắp xếp thứ tự theo tên sách. Trái ngược với thành phần Details chỉ hiển thị một thực thể tại một thời điểm, Multiple Details hiển thị tất cả các đối tượng thỏa mãn biểu thức điều kiện.

Multiple Book Details

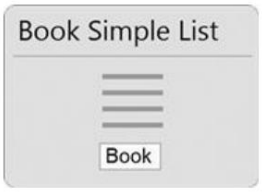
isbn	title	authorsName	price	pageCount	coverImage	quantityInStock
0671746723	Dirk Gently's Holistic Detective Agency	Douglas N. Adams	6.99	306		12
0671742515	The Long Dark Tea-Time of the Soul	Douglas N. Adams	6.99	307		4
0517149257	The Ultimate Hitchhikers Guide	Douglas N. Adams	122	815		4

Hình 2.8. Giao diện cho thành phần Multiple Details.

Simple List: trình bày một hoặc nhiều thuộc tính của một tập các thực thể của một lớp dưới dạng danh sách.

Mặc dù Multipel Details thường được sử dụng để biểu diễn chi tiết của nhiều đối tượng trong cùng một trang, nhưng một Simple List và các biến thể của nó được sử dụng khi cần một danh sách hoặc menu để người dùng chọn một hoặc nhiều phần tử và thực hiện các hành động với chúng. Các thuộc tính cần thiết của Simple List cũng giống như các thuộc tính của Multiple Details.

Ví dụ dưới đây trình bày Simple List nhằm chọn sách dựa trên tên sách. Hình 2.9 trình bày mô hình cho thành phần Simple List các tên sách và Hình 2.10 cho thấy giao diện tương ứng.



Hình 2.9. Thành phần Simple List.

Hình 2.10. Giao diện của Simple List.

List: phiên bản nâng cao của Simple List mà cho phép cuộn các phần tử và sắp xếp lại chúng bằng một cú nhấp chuột đơn giản. Bên cạnh các thuộc tính đã được xác định cho Simple List, List bao gồm thêm các thuộc tính cụ thể sau:

- Thuộc tính sắp xếp (Sortable): Là một thuộc tính kiểu boolean mà khi thuộc tính này bằng true thì danh sách có thể được sắp xếp động.
- Thuộc tính sắp xếp mặc định (Default sort attributes): Nếu danh sách là sortable, nó xác định thuộc tính sắp xếp được người dùng chọn để sắp xếp cho danh sách đó và đó cũng là tiêu chí sắp xếp mặc định khi danh sách được hiển thị lần đầu tiên.
- Kích thước lịch sử sắp xếp (Sort history size): Chỉ định số lượng hành động sắp xếp của người dùng phải được danh sách ghi nhớ.
- Khả năng kiểm tra (Checkable): Nếu thuộc tính này bằng true, một check box sẽ được hiển thị cho từng phần tử của danh sách.
- Yếu tố khối (Block factor): Chỉ định số lượng phần tử được hiển thị tại một thời điểm. Nếu thuộc tính này không xác định hoặc có giá trị -1, tất cả các phần tử đều được hiển thị. Nếu hệ số khối được xác định, thì các thanh cuộn sẽ được hiển thị cho danh sách.

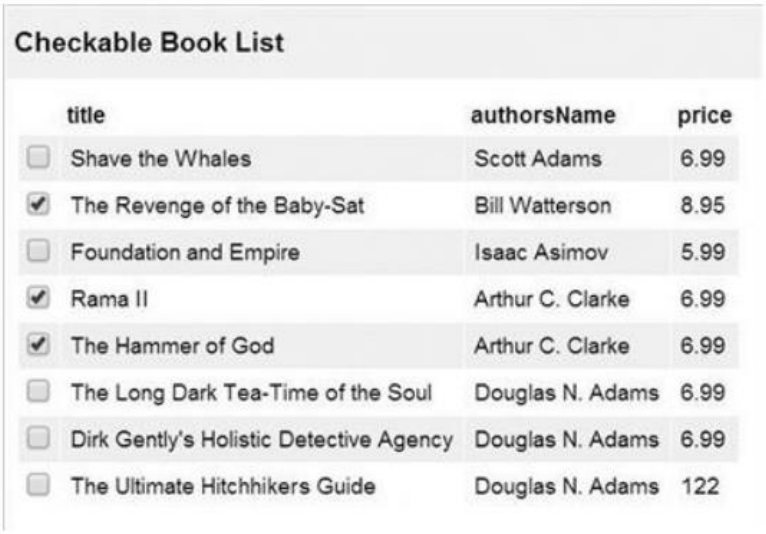
Hình 2.11 trình bày thành phần IFML cho một List. Danh sách này được xác định cho lớp Book với ba thuộc tính: tên tác giả, tên sách và giá tiền. Yếu tố khối trong trường hợp này được định nghĩa bằng 3. Hình 2.12 biểu diễn giao diện tương ứng của mô hình List trong Hình 2.11.

Hình 2.11. Thành phần List.

Hình 2.12. Giao diện cho thành phần List.

Checkable List: giống với Simple List nhưng cho phép người dùng chọn một tập hợp các phần tử từ danh sách. Hình 2.13 trình bày định nghĩa IFML của một Checkable List và Hình 2.14 trình bày giao diện tương ứng của nó. Trong khi Simple List chỉ cho phép chọn một đối tượng tại một thời điểm, Checkable List cho phép chọn bất kỳ số lượng đối tượng nào.

Hình 2.13. Thành phần Checkable List.



Hình 2.14. Giao diện của Checkable List.

Form: được sử dụng để nhập dữ liệu. Nó rất hữu ích để tạo các thực thể mới. Form cũng có thể dùng để cung cấp các tham số cho các tìm kiếm, truy vấn và lệnh.

Form bao gồm một tập hợp các trường (field). Mỗi trường chứa một giá trị chữ hoặc số. Có các trường văn bản, trường lựa chọn và trường đa lựa chọn. Form cũng có thể liên kết với một lớp. Nếu nó được liên kết với một lớp, thì các trường của nó có thể được liên kết với các thuộc tính của một lớp.

Trong trường hợp Form được sử dụng để thu thập các tham số cho một truy vấn hoặc lệnh thường không cần phải liên kết với các lớp. Ví dụ: form thu thập từ khóa để tìm kiếm sách không cần phải liên kết với bất kỳ lớp nào.

Các trường của một form có các thuộc tính sau:

- Tên (Name): Tên của trường do người thiết kế xác định
- Thuộc tính (Attribute): Nếu Form được liên kết với một lớp, thì trường có thể được liên kết với một trong các thuộc tính của lớp đó.
- Phân loại nội dung (Content type): Xác định phân loại kiểu dữ liệu của trường, có thể là String, text, blob hoặc url.
- Tải trước (Preload): Trường có thể được tải trước khi người dùng yêu cầu, các giá trị được tải trước này được hiển thị cho người dùng khi form được hiển thị.
- Sự thay đổi (Modifiable): Xác định xem giá trị ban đầu của trường có thể bị thay đổi hay không.

Hình 2.15 trình bày một Form được liên kết với lớp Book và hình 2.16 trình bày giao diện của nó.

Hình 2.15. Thành phần Form.

Hình 2.16. Giao diện của thành phần Form

Hierarchies: Được sử dụng để hiển thị các đối tượng phụ thuộc có hai cấp trở lên. Hệ thống phân cấp Hierarchies dùng để hiển thị các đối tượng có liên kết một nhiều như Publisher và Book, hoặc các đối tượng liên kết toàn bộ như Book và Chapter. Hình 2.17 cho thấy định nghĩa của thành phần hệ thống phân cấp Hierarchies hai cấp với Publisher ở cấp một và Book ở cấp thứ hai.

Hình 2.17. Thành phần phân cấp Hierarchies.

Mức thứ hai thường xác định một biểu thức điều kiện dựa trên vai trò. Trong ví dụ này, biểu thức điều kiện dựa trên vai trò xác định rằng chỉ những Book được liên kết với Pulisher nhất định mới được hiển thị bên dưới nó.

Hình 12.18 cho thấy giao diện cho hệ thống phân cấp Hierarchies của Hình 12.17 với các Book được liên kết với Publisher tương ứng của nó.

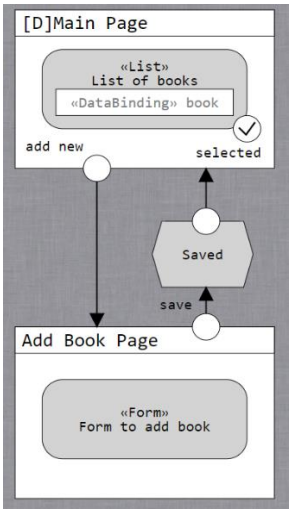
Hình 2.18. Giao diện hệ thống phân cấp Hierarchies.

2.3.2.3. Thành phần Event và Action

Event được liên kết với View Container hoặc View Component cho biết tương tác của người dùng với các phần tử, hay còn được gọi là View Element Event. View Element Event được sở hữu bởi các phần tử View liên quan của chúng. Điều này nghĩa là View Elements chứa các Event cho phép người dùng kích hoạt một tương tác trong ứng dụng, ví dụ như click vào một link url hoặc một Button. Có thể nói rằng Event đại diện cho sự kiện tương tác của người dùng, có thể được kích hoạt bởi View Element (View Container và View Component).

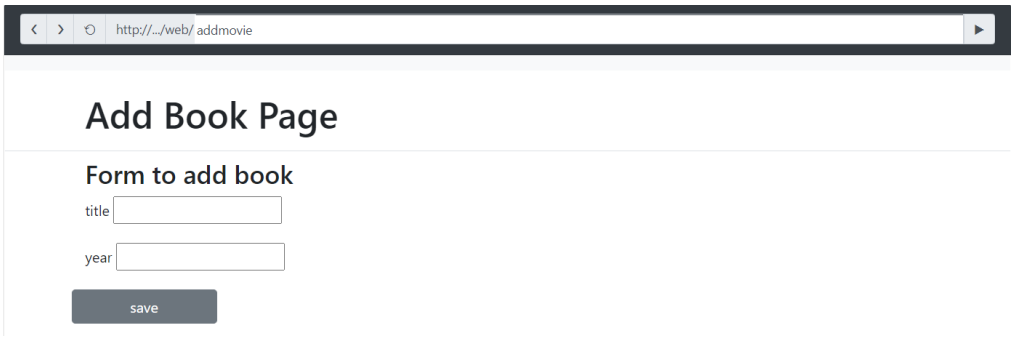
Action là một thành phần tương ứng với sự thay đổi trạng thái của giao diện gây ra bởi hành vi của người dùng. Một Action cần phải đi kèm với Event. Các hành động được thực hiện trước khi thay đổi trạng thái của giao diện. Các hành động có thể chứa các chức năng thông thường như là cập nhật, thêm hoặc xóa.

Hình 2.19 trình bày một ví dụ về việc thêm một book mới vào danh sách. Người dùng nhập thông tin của book vào form để thêm mới và ấn save để lưu. Việc người dùng click vào button save và chuyển sang giao diện BookList được coi là một event. Sau khi kích hoạt event save, một phương thức(function) sẽ được gọi và thực hiện việc lấy các thông tin của book mà người dùng điền trong form và lưu các dữ liệu đó vào cơ sở dữ liệu để hiển thị chúng lên danh sách. Việc gọi phương thức lưu trữ dữ liệu book mới đó được thực hiện thông qua Action saved.



Hình 2.19. Action chuyển đổi dữ liệu từ Form sang List.

Hình 2.20 trình bày giao diện tương ứng thêm một book mới vào danh sách.



Hình 2.20. Giao diện cho Action chuyển đổi dữ liệu từ Form sang List.

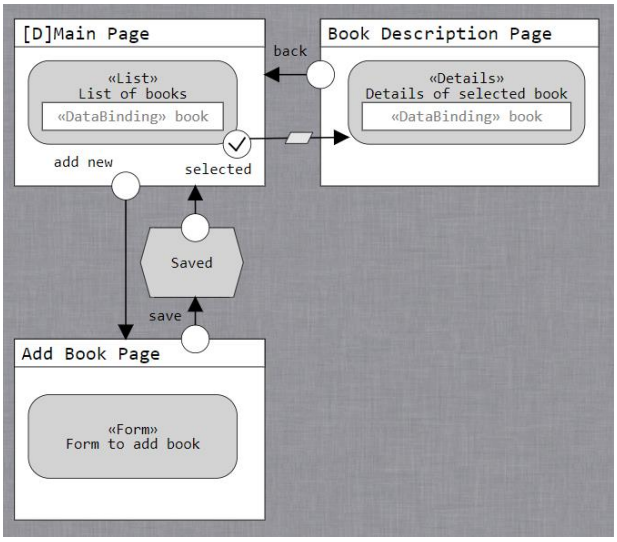
Một Action luôn đi kèm với một Event, nhưng một Event có thể có hoặc không có Action đi kèm. Một vài thao tác của người dùng như click vào một url, click button back, button home là các event mà không cần action đi kèm. Các action thường đi kèm với các Event liên quan đến việc thao tác tới cơ sở dữ liệu và thay đổi trạng thái giao diện như thêm, xóa, cập nhật.

Để hiểu thêm về Action, cần tìm hiểu về Parameter Binding. Parameter Binding là mối quan hệ được hình thành giữa thành phần UI và Action. Parameter Binding là một phần thông tin được truyền từ thành phần giao diện gốc đến thành phần giao diện đích của một action. Một parameter binding có tên và giá trị, được lấy tại thành phần gốc của nó.

Các khóa đối tượng hoặc giá trị thuộc tính thường được truyền qua Parameter Binding. Một Action của một View Component liên kết khóa của đối tượng đã chọn tại thành phần gốc và đưa nó đến đích bằng một biểu thức điều kiện để có thể được sử dụng nó. Với các Action và Parameter Binding, một View Component có thể xác định các phần tử nào được hiển thị trong ViewComponent khác.

2.3.3. Ví dụ minh họa mô hình IFML

Hình 2.21 trình bày một ví dụ minh họa về mô hình IFML mô tả chương trình quản lý Book.



Hình 2.21. Mô hình IFML mô tả chương trình quản lý Book.

Mô hình IFML mô tả chương trình quản lý movie bao gồm ba thành phần View Container Main Page, Add Book Page, Book Description Page. Main Page chứa một thành phần View Component dạng List, hiển thị danh sách các book, Book Description Page chứa thành phần Details mô tả chi tiết của một book và Add Book Page chứa một thành phần Form để điền thông tin một book mới mà muốn thêm vào danh sách. Mô hình bao gồm bốn Event là select, add new, back và save và một Action save. Khi người dùng muốn xem thông tin chi tiết của một book, event select được thực hiện để chuyển đổi sang Book Description Page để hiển thị thông tin chi tiết của book được chọn. Tương tự đối với Event add new, event được gọi khi người dùng muốn thêm một book mới vào danh sách, Event add new sẽ chuyển từ Main Page sang Add Book Page hiển thị một Form để người dùng có thể nhập thông tin của book mới muốn thêm vào. Cuối cùng sau khi nhập thông tin book mới, action save được thực hiện để lưu thông tin book mới được thêm vào danh sách book và hiển thị ở List of Book trong Main Page.

2.4. Đặc tả yêu cầu chức năng

Luận văn sử dụng ba đặc tả yêu cầu đầu vào để sinh tự động giao diện người dùng. Đó là đặc tả yêu cầu chức năng của giao diện người dùng viết bằng ngôn ngữ tự nhiên, sơ đồ lớp dạng văn bản và sơ đồ ca sử dụng dạng văn bản.

2.4.1. Đặc tả yêu cầu chức năng của giao diện người dùng bằng ngôn ngữ tự nhiên

Yêu cầu được đưa ra từ người dùng được viết bằng ngôn ngữ tự nhiên và tiếng Anh. Các yêu cầu bằng ngôn ngữ bình thường mang nhiều tạp âm, mơ hồ, mâu thuẫn và khó phân tích. Điều này gây ra nhiều thời gian xử lý hơn và tạo ra kết quả lỗi trong trường hợp dữ liệu phức tạp. Do đó, để khắc phục những vấn đề này, luận văn đã sử dụng một tập luật chuyển đổi để thu thập đủ thông tin từ đặc tả yêu cầu. Đặc biệt, luận văn cũng áp dụng quy tắc để viết được các yêu cầu về giao diện người dùng trong văn bản ngôn ngữ tự nhiên phù hợp [10].

Các quy tắc cho việc viết đặc tả giao diện người dùng bằng ngôn ngữ tự nhiên:

- Các câu cần ngắn gọn và súc tích. Việc tách các câu ghép tạo ra ít mơ hồ hơn và tránh mất thông tin.
- Nên viết yêu cầu bằng những câu chủ động và đơn giản.
- Các danh từ riêng và danh từ số nhiều trong câu nên được chuyển đổi thành danh từ số ít. Điều này giúp xác định các thành phần chính của câu danh từ, động từ, tính từ để sử dụng trong các luật chuyển đổi.
- Yêu cầu không được bao gồm các câu phi chức năng và phủ định.
- Các câu trong yêu câu không nên sử dụng từ viết tắt như e.g, hay các dấu ngoặc để tránh sai sót cho việc gắn thẻ POS cho các từ trong câu.
- Yêu cầu phải được viết từ quan điểm của người dùng, mô tả những chức năng của chương trình mà người dùng mong muốn được hiển thị trong giao diện.
- Yêu cầu của giao diện người dùng nên được viết lần lượt theo từng trang giao diện (view by view).
- Yêu cầu nên bao gồm các từ khóa (keyword) được sử dụng phổ biến khi mô tả UI như page, list, form, detail, button, …

Dưới đây là một ví dụ về đặc tả yêu cầu chức năng bằng ngôn ngữ tự nhiên cho một ứng dụng Book Store Online:

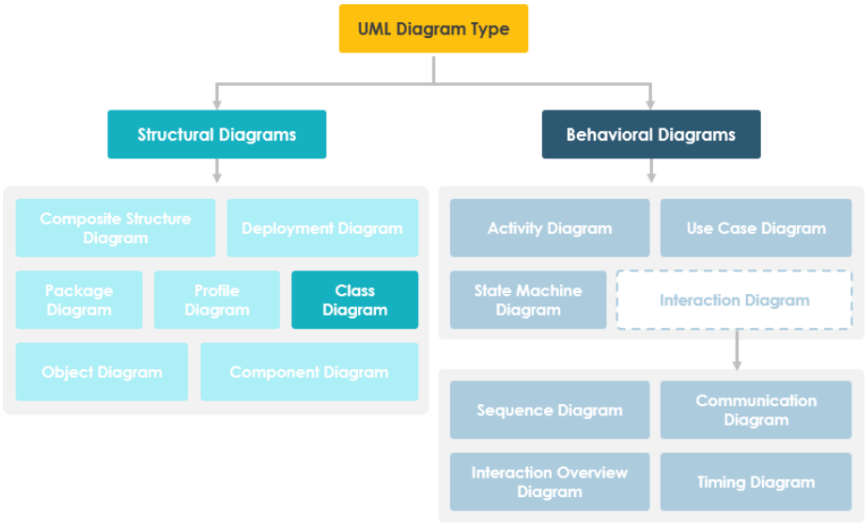
"The home page of the online Bookstore present a list of category consisting of book, recording and software. In this sub clause, the user can select one of the product category, or go directly to the shopping cart. After selecting a category, user is directed to the product page that display a list of product. When the user select a product, obtain the detail of selected item along with the option to add the product to the cart. Once the user press "add to cart" button, a quantity window appear which present a form to add quantity of product. After accrpting the quantity, the article is added to the cart, and a confirmation window is displayed. The shopping cart preview the list of product selected by the user with a checkout button. In this sub clause, are shown the quantity and the order detail. The user is able to update the cart by changing the quantity, empty the cart by deleting all the product of the current order, and start the payment process by clicking in checkout button. When the user choose to update the cart, the total amount is recalculated.

When the user empty the cart is redirected to a confirm page. The checkout button navigate to the customer infomation page that display a form to input personal infomation. He must provide his personal information and press "Next" button. After providinig his personal information, the user must provide his bank account information and confirm the payment in order to proceed with the transaction where payment is executed using "pay" button. After performing the transaction, a confirm page appear that shows details of payment."

Đặc tả yêu cầu chức năng bằng ngôn ngữ tự nhiên dạng văn bản này có thể áp dụng các luật chuyển đổi trong mục 3.2 để sinh các thành phần của mô IFML mô tả giao diện người dùng. Sau đó kết hợp với sơ đồ lớp và sơ đồ ca sử dụng sinh bản mẫu giao diện người dùng.

2.4.2. Sơ đồ lớp dạng văn bản

Sơ đồ lớp (Class diagram): là một loại sơ đồ UML cấu trúc tĩnh mô tả cấu trúc của hệ thống bằng cách hiển thị các lớp của hệ thống, thuộc tính, hoạt động (hoặc phương thức) của chúng và mối quan hệ giữa các đối tượng.



Hình 2.22. Sơ đồ lớp là một loại sơ đồ UML cấu trúc tĩnh.

Một sơ đồ lớp được tạo thành từ hai phần:

- Tập hợp các lớp.
- Tập hợp các mối quan hệ giữa các lớp.

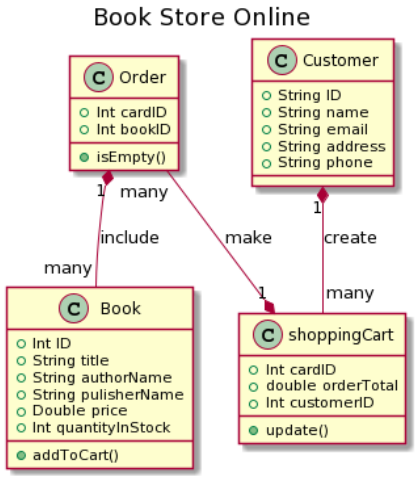
Lớp (class): mô tả một nhóm đối tượng có vai trò giống nhau trong hệ thống. Một lớp được biểu diễn trong sơ đồ bao gồm ba thành phần:

- Tên lớp
- Thuộc tính (attribute): là các mô tả về cấu trúc hoặc tính năng tĩnh của một lớp, các thuộc tính được ánh xạ thành các biến thành viên trong code.

- Phương thức (method): là một tính năng hành vi, xác định những đối tượng của lớp có thể làm gì. Các phép toán ánh xạ phương thức lên các hàm thực thi trong code.

Mối quan hệ (relationship): là các loại kết nối logic cụ thể giữa các lớp đối tượng. Một lớp có thể tham gia vào một hoặc nhiều mối quan hệ với các lớp khác. Mối quan hệ được chia thành 4 loại: Thừa kế (Inheritance), Kết hợp (Aggregation), Thành phần (Composition) và Phụ thuộc (Dependency).

Một sơ đồ lớp có thể được biểu diễn bằng dạng mô hình UML trực quan hoặc dạng văn bản. Hình 2.23 trình bày một ví dụ về sơ đồ lớp dạng mô hình UML cho ứng dụng quản lý cửa hàng sách (Book Store Online) và Hình 2.24 trình bày dạng văn bản tương ứng của nó.



Hình 2.23. Sơ đồ lớp dạng mô hình cho ứng dụng Book Store Online.

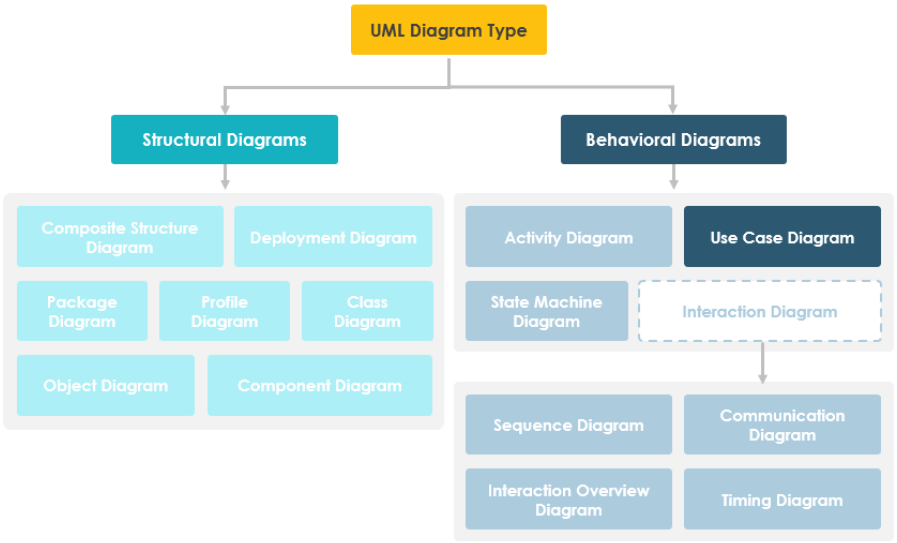
```
@startuml

title Book Store Online

class Book {
  +Int ID
  +String title
  +String authorName
  +String pulisherName
  +Double price
  +Int quantityInStock
  +addToCart()
}

class Order{
  +Int cardID
  +Int bookID
  +isEmpty()
}

class shoppingCart{
  +Int cardID
  +double orderTotal
  +Int customerID
  +update()
}

class Customer{
  +String ID
  +String name
  +String email
  +String address
  +String phone
}

Customer "1" *-down- "many" shoppingCart: create
shoppingCart "1" *-up- "many" Order: make
Order "1" *-down- "many" Book: include

@enduml
```

Hình 2.24.Sơ đồ lớp Book Store Online biểu diễn dạng văn bản.

Luận văn sử dụng sơ đồ lớp dạng văn bản làm một trong các đặc tả yêu cầu đầu vào để sinh tự động các đối tượng của chương trình và các thuộc tính của đối tượng được hiển thị trong giao diện người dùng.

2.4.3. Sơ đồ ca sử dụng dạng văn bản

Sơ đồ ca sử dụng (Use case diagram): là một loại sơ đồ UML hành vi giúp thiết kế một hệ thống từ quan điểm của người dùng cuối. Nó có thể chỉ định tất cả các hành vi hệ thống mà nhìn thấy được.

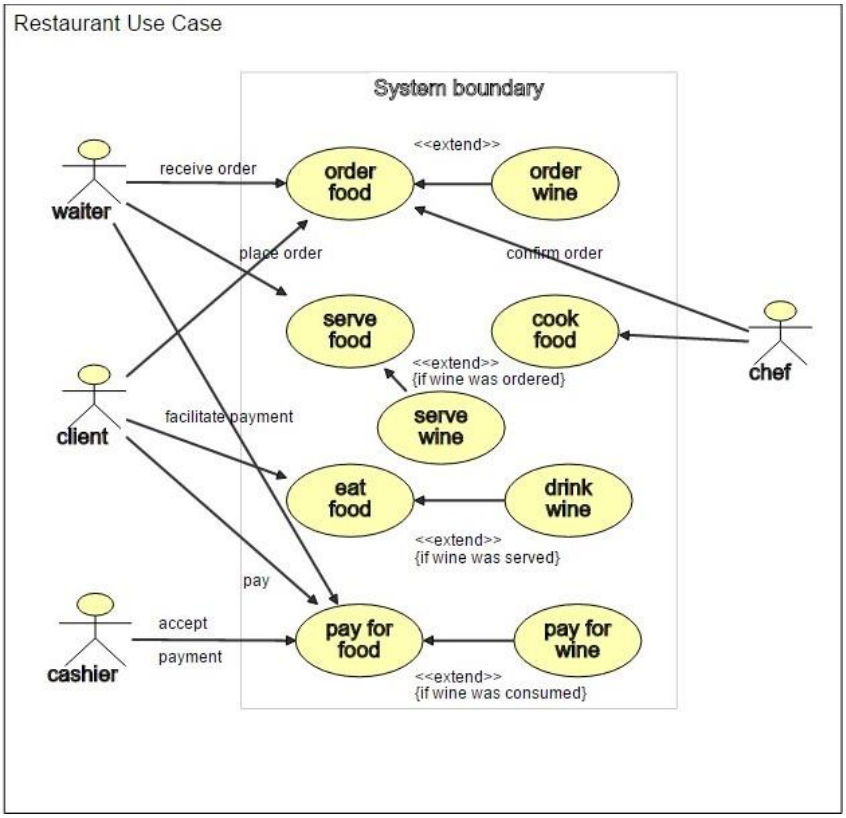


Hình 2.25. Sơ đồ ca sử dụng là một loại sơ đồ UML hành vi.

Một biểu đồ ca sử dụng tiêu chuẩn được định nghĩa trong UML bao gồm các khái niệm sau:

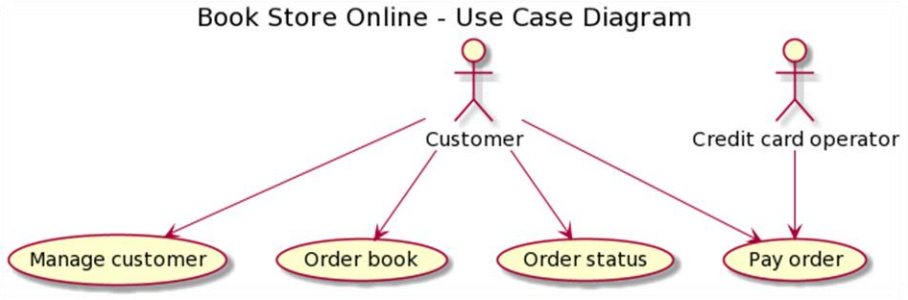
- Tác nhân (actor): tác nhân đóng vai trò tương tác với các ca sử dụng
- Ca sử dụng (use case): mô tả các chức năng của hệ thống (quy trình tự động hoặc thủ công). Mỗi tác nhân phải được liên kết với một ca sử dụng, trong khi một số ca sử dụng có thể không cần liên kết với các tác nhân.
- Liên kết (association): nhằm kết nối tác nhân với ca sử dụng
- Ranh giới hệ thống (system boundary): có thể là toàn bộ hệ thống đối với các hệ thống nhỏ. Đối với các hệ thống lớn và phức tạp, ranh giới hệ thống có thể là một phần (module).

Hình 2.26 trình bày một ví dụ về sơ đồ ca sử dụng bao gồm các thành phần actor, usecase, association và system boundary.



Hình 2.26. Các thành phần của sơ đồ ca sử dụng.

Sơ đồ ca sử dụng có thể được biểu diễn dưới hai dạng mô hình UML trực quan và dạng văn bản. Một ví dụ về sơ đồ ca sử dụng cho ứng dụng quản lý cứ hàng sách Book Store Online được chỉ ra bên dưới. Hình 2.27 mô tả sơ đồ ca sử dụng biểu diễn dưới dạng mô hình UML trực quan và Hình 2.28 mô tả sơ đồ ca sử dụng dạng văn bản.



Hình 2.27. Sơ đồ ca sử dụng biểu diễn dưới dạng mô hình UML.

```
@startuml

title Book Store Online - Use Case Diagram

:Customer:
:Credit card operator:
usecase (Manage customer)
usecase (Order book)
usecase (Order status)
usecase (Pay order)

Customer --> (Manage customer)
Customer --> (Order book)
Customer --> (Order status)
Customer --> (Pay order)
:Credit card operator: --> (Pay order)

@enduml
```

Hình 2.28. Sơ đồ ca sử dụng biểu diễn dưới dạng văn bản.

Sơ đồ ca sử dụng trong đặc tả yêu cầu ngoài việc biểu diễn các chức năng chính của phần mềm còn thể hiện được mối quan hệ giữa các chức năng của phần mềm. Do dó, nó được sử dụng để trích xuất các thông tin cho các liên kết giữa các thành phần của mô hình IFML mô tả giao diện người dùng.

2.4. Thư viện xử lý ngôn ngữ tự nhiên OpenNLP

OpenNLP là một thư viện mã nguồn mở java được sử dụng sử lý ngôn ngữ tự nhiên dạng văn bản. OpenNLP cung cấp các dịch vụ như mã hóa (tokenization), phân đoạn câu (sentence segmentation), gắn thẻ một phần (part-of-speed tagging), trích xuất thực thể (entity extraction), phân tích cú pháp (parsing) [9].

Luận văn xử lý đặc tả yêu cầu chức năng dạng văn bản bằng cách áp dụng các phương pháp xử lý ngôn ngữ tự nhiên sau:

- Tách câu (Sentence Detection): phát hiện các câu trong văn bằng cách sử dụng biểu thức chính quy và một tập hợp các quy tắc đơn giản. Nó chuyển một tệp văn bản dài thành tập hợp các câu để giảm độ phức tạp của tệp văn bản.
- Mã hóa (Tokenization): chuyển đổi câu và tách thành các mã (token). Các mã token có thể là từ, số hoặc dấu câu.

- Gắn thẻ POS (POS Tagging): cung cấp khả năng gán mỗi từ của câu đã cho với một POS tương ứng, xác định từ loại của từ như noun, verb, adverd, adjective,...

Bảng 2.1 bên dưới, mô tả danh sách các ký hiệu mô tả các từ khi áp dụng kỹ thuật gắn thẻ POS.

Bảng 2.1. Danh sách các ký hiệu sử dụng trong POS Tagging

Sr.#	Tag	Description	Examples
1	NN	Noun in Singular Form	Movie
4	NNP	Proper Noun in Singular Form	Human
2	NNS	Noun in Plural Form	Movies
3	NNPS	A proper noun, plural	Journal
5	TO	To	To
6	CC	coordinating conjunction	And, but, or,
7	IN	preposition/subordinating conjunction	By, of, against, with, for, in, while, on, from, that, along, until, if, though, as, at, into, about, than, during, before, since
8	PRP$	possessive pronoun	Its, their, his, her,
9	DT	Determiner	The, a, an, both, each, Any, all, this
10	WDT	wh-determiner	Which, that
11	JJ	Adjective,	Beautiful, nice
12	JJR	adjective, comparative	Taller, smaller
13	VB	verb, the base form	Live, fight
14	VBN	verb, past participle	Looked, booked
15	VBZ	verb, 3rd person sing. present	Climbs, drinks
16	MD	Modal	Must, will, may, should, can, cannot
17	CD	cardinal number	Two, one, four, ten, three

2.5. Tổng kết chương

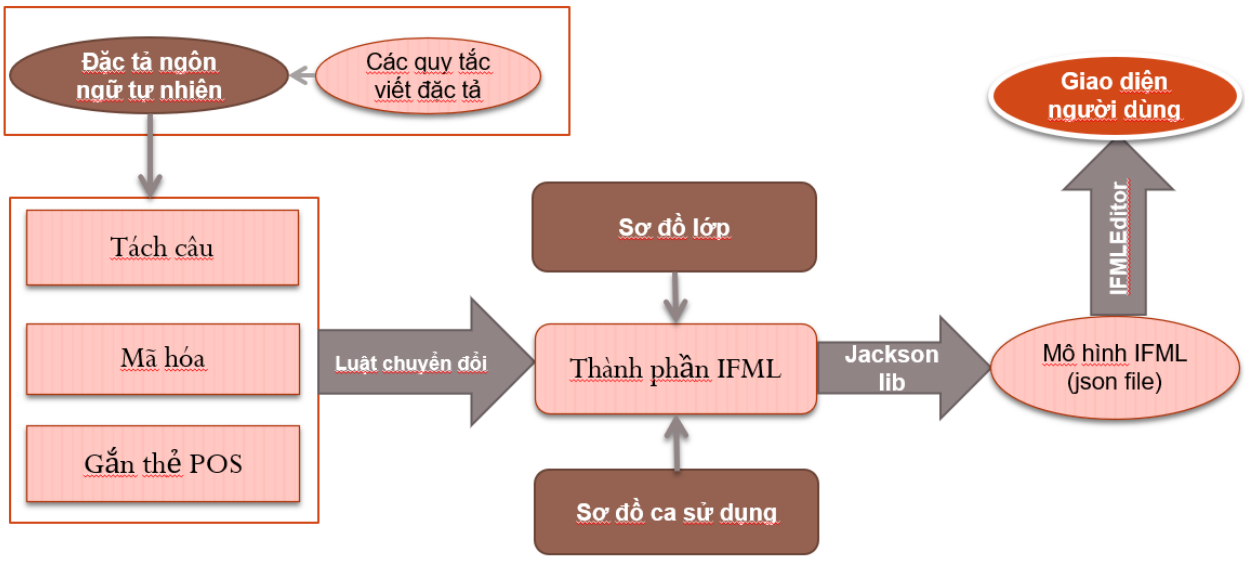
Các cơ sở lý thuyết và kiến thức nền tảng cần thiết cho luận văn đã được giới thiệu trong chương vừa rồi. Các khái niệm, vai trò và các cấp độ mô hình hóa của kỹ nghệ hướng mô hình MDE đã được trình bày ở đầu chương. Chương này cũng đã tìm hiểu về ngôn ngữ mô hình hóa dựa trên MDE, cụ thể là ngôn ngữ mô hình hóa luồng tương tác IFML, bao gồm khái niệm cũng như cú pháp và ngữ nghĩa của các thành phần IFML. Luận văn cũng trình bày kiến thức cơ bản về các đặc tả yêu cầu chức năng được sử dụng để sinh tự động giao diện người dùng. Đồng thời, tìm hiểu về thư viện xử lý ngôn ngữ tự nhiên OpenNLP nhằm xử lý các đặc tả yêu cầu đầu vào trước đó.

CHƯƠNG 3. SINH TỰ ĐỘNG GIAO DIỆN NGƯỜI DÙNG TỪ ĐẶC TẢ YÊU CẦU CHỨC NĂNG

Chương này trình bày về quy trình sinh tự động bản mẫu giao diện người dùng từ các đặc tả yêu cầu chức năng thông qua việc sinh ra json file biểu diễn mô hình IFML nhờ các tập luật chuyển đổi. Sau đó, sử dụng công cụ hỗ trợ IFML Editor để sinh ra giao diện người dùng tương ứng đáp ứng các yêu cầu của người dùng trong đặc tả yêu cầu.

3.1. Giới thiệu

Trong chương này, ở phần đầu chương, luận văn sẽ trình bày về các tập luật chuyển đổi sinh ra các thành phần IFML. Sau đó, kết hợp các thành phần IFML được sinh ra với các mô hình ca sử dụng dạng văn bản và mô hình lớp dạng văn bản để sinh ra json file biểu diễn một mô hình IFML. Cuối cùng là sử dụng công cụ hỗ trợ IFML Editor để sinh tự động bản mẫu giao diện người dùng từ mô hình IFML. Hình 3.1 trình bày sơ đồ mô tả phương pháp sinh tự động bản mẫu giao diện người dùng từ đặc tả yêu cầu chức năng mà được nghiên cứu trong luận văn này.



Hình 3.1. Sơ đồ mô tả phương pháp sinh giao diện người dùng.

3.2. Các luật chuyển đổi xác định thành phần cấu trúc IFML

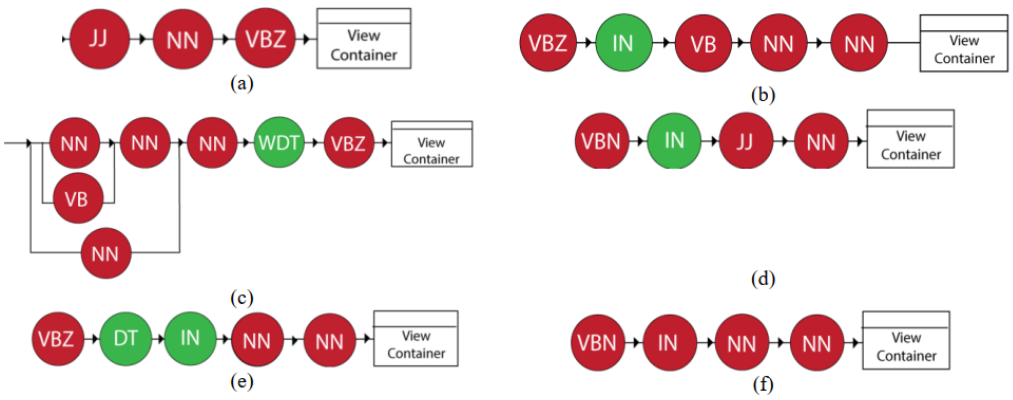
Phần này trình bày các luật chuyển đổi để trích xuất các thành phần IFML từ đặc tả ngôn ngữ tự nhiên và bằng tiếng Anh. Các đặc tả về ngôn ngữ tự nhiên này cần tuân theo các quy tắc đã được chỉ ra trong mục 2.4.1. Theo [10] các luật chuyển đổi này xác định các câu trong ngôn ngữ tự nhiên tương ứng với các thành phần IFML. Phương pháp sử dụng kỹ thuật xử lý ngôn ngữ tự nhiên POS Tagging để xác định các thẻ (tag) của câu, sau đó so sánh nó với các tập luật để xác định thành phần IFML tương tứng.

Luận văn tập trung trình bày các luật chuyển đổi để xác định bốn thành phần IFML cơ bản sau:

- View Container
- View Component
- Event
- Action

3.2.1. Luật chuyển đổi cho View Container

Có sáu quy tắc xác định View Containers. Hình 3.9 minh họa cho các luật chuyển đổi cho View Container (RC - Rule for ViewContainer), trong đó vòng tròn màu đỏ là điều bắt buộc phải có để áp dụng quy tắc, và vòng tròn màu xanh lá là điều tùy chọn, có thể có hoặc không.



Hình 3.2. Các luật chuyển đổi cho View Container.

RC1: Tính từ JJ và danh từ NN cùng nhau tạo thành Cụm danh từ, với động từ ở ngôi thứ ba số ít VBZ trong một câu, sự kết hợp này tạo thành một View Container như trong Hình 3.9(a).

Ví dụ: A main page exhibits list of movies

Luật chuyển đổi: \w+/JJ \w+/NN \w+/VBZ

RC2: Một động từ VBZ đi với giới từ tùy ý cộng với một VB (động từ dạng cơ sở) trước hai danh từ, tạo thành Cụm động từ trong một câu, sự kết hợp này tạo thành một View Container như trong Hình 3.9(b).

Ví dụ: Furthermore, the home page facilitates user to add a new books through add new button which navigates to add books page.

Luật chuyển đổi: (\w+/VBZ (\w+/IN |to/TO)?\w+/VB \w+/NN \w+/NN)

RC3: Quy tắc này là sự kết hợp của ba quy tắc chia sẻ các đặc điểm chung, do đó chúng được hợp nhất thành một để dễ hiểu như thể hiện trong Hình 3.9(c).

R3.1: Nếu hai thẻ NN liên tiếp theo sau bởi một VBZ, nó có thể tạo thành một View Container . Nó cũng có thể bao gồm thêm một thẻ WDT tùy chọn trước VBZ trong câu.

R3.2: Một thẻ NN được theo sau bởi các thẻ NN liên tiếp cùng với một động từ VBZ được coi là một ViewContainer. Nó cũng có thể bao gồm thêm một thẻ WDT tùy chọn trước VBZ trong câu.

R3.3: Nếu một thẻ VB được theo sau bởi hai thẻ NN liên tiếp cùng với một VBZ, nó được coi là ViewContainer. Nó cũng có thể bao gồm thêm một thẻ WDT tùy chọn trước VBZ trong câu.

Ví dụ: A confirmation page is displayed after successful addition.

Luật chuyển đổi: ((\w+/NN\w+/NN(\w+/WDT)?\w+/VBZ)|((\w+/NN|\w+/VB)\w+/NN \w+/NN (\w+/WDT)?\w+/VBZ))

RC4: Một động từ VBN với giới từ IN tùy chọn, theo sau là tính từ JJ và Danh từ NN tạo thành một Cụm danh từ trong câu, sự kết hợp này có thể coi là một View Container như trong Hình 3.9(d).

Ví dụ: A user can select desired movie from the list of movie presented on main page in order to view its detail.

Luật chuyển đổi: (\w+/VBN (\w+/IN |to/TO)?\w+/JJ \w+/NN)

RC5: Một động từ VBZ với giới từ IN tùy chọn, theo sau là hai thẻ NN tạo thành một ViewContainer như trong Hình 3.9(e).

Ví dụ: In order to view products from a specific category, a user can “select” one of the categories, which directs to products page.

Luật chuyển đổi: (\w+/VBZ(\w+/DT)?(\w+/IN |to/TO)?\w+/NN \w+/NN)

RC6: Một động từ dạng VBN với giới từ IN, theo sau là hai thẻ NN liên tiếp tạo thành ViewContainer như trong Hình 3.9(f).

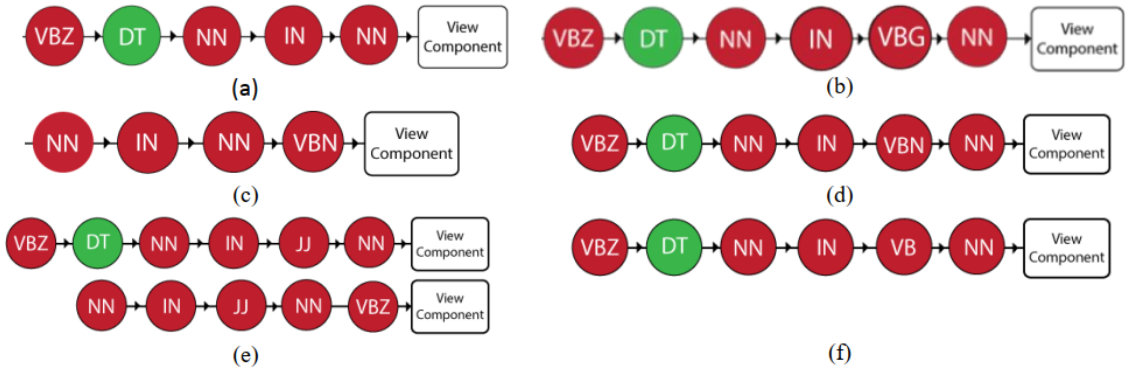
Ví dụ: Finally user is redirected to confirmation page.

Luật chuyển đổi: (\w+/VBN(\w+/IN |to/TO)\w+/NN \w+/NN)

3.2.2. Luật chuyển đổi cho View Component

Có sáu quy tắc để trích xuất các View Component từ đặc tả yêu cầu dạng văn bản. Hình 3.10 trình bày các luật chuyển đổi cho View Component (RP - Rule for View

Component) , trong đó vòng tròn màu đỏ thể hiện tính năng bắt buộc phải có để áp dụng luật chuyển đổi và vòng tròn màu xanh biểu thị tính năng tùy chọn, có thể có hoặc không.



Hình 3.3. Các luật chuyển đổi cho View Component.

RP1: Một động từ dạng VBZ theo sau bởi một DT, sau đó kết hợp với thẻ danh từ NN, giới từ IN và tiếp tục là thẻ danh từ NN trong một câu sẽ tạo thành một View Component. DT là thành phần tùy chọn như trong Hình 3.10 (a).

Ví dụ: A main page exhibits a list of movies.

Luật chuyển đổi: (\w+/VBZ (\w+/DT)?\w+/NN (\w+/IN |\w+/TO)\w+/NN)

RP 2: Luật chuyển đổi bao gồm một động từ VBZ ở đầu câu, sau đó kết hợp với thẻ DT, Danh từ NN, giới từ IN và các thẻ VBG, NN trong cùng một câu, trong đó DT là tùy chọn như trong Hình 3.10(b).

Ví dụ: The home page contains a form for adding movie details, i.e. title of movie, and year of release.

Luật chuyển đổi: (\w+/VBZ (\w+/DT)? \w+/NN \w+/IN \w+/VBG \w+/NN)

RP 3: Sự kết hợp của danh từ NN với giới từ IN và các thẻ NN, VBN có thể trở thành một thành phần View Component như trong Hình 3.10(c)

Ví dụ: The user can view details of each complaint from list of complaints, presented by home page.

Luật chuyển đổi: (\w+/NN \w+/IN \w+/NN \w+/VBN)

RP 4: Luật chuyển đổi bao gồm một động từ dạng VBZ, sau đó kết hợp với thẻ DT, Danh từ NN, giới từ IN và thẻ VBN, NN trong cùng một câu, trong đó DT là tùy chọn như trong Hình 3.10(d)

Ví dụ: It leads to movie detail page which presents detail of selected movie; here “Selected” which is a VBN works as adjective.

Luật chuyển đổi: (\w+/VBZ(\w+/DT)?\w+/NN \w+/IN \w+/VBN \w+/NN)

RP 5: Quy tắc này bao gồm hai quy tắc phụ như trong Hình 3.10(e), đó là:

R5.1: Một động từ VBZ được theo sau bởi sự kết hợp của danh từ NN, giới từ IN và cụm danh từ [JJ-NN] tạo thành một View Component.

Ví dụ: The page of shopping cart previews a list of selected products with a “checkout button”.

R5.2: Sự kết hợp của danh từ NN, giới từ IN và một cụm danh từ [JJ-NN] theo sau bởi VBZ có thể tạo thành một View Component.

Ví dụ: This page contains a list of products. On “selecting” a product, the detail of selected product appears along with an option to add product to the cart.

Luật chuyển đổi: ((\w+/VBZ(\w+/DT) ?\w+/NN (\w+/IN |\w+/TO)\w+/JJ\w+/NN)| (\w+/NN (\w+/IN |\w+/TO)\w+/JJ\w+/NN \w+/VBZ))

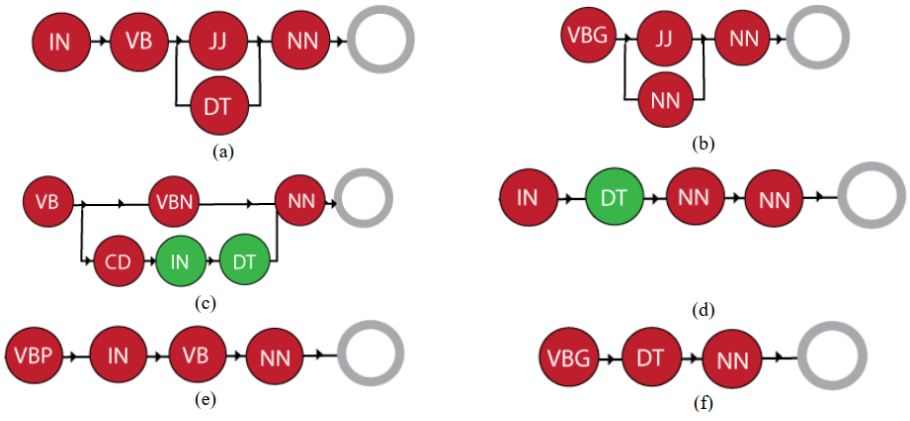
RP 6: Luật chuyển đổi bao gồm một động từ VBZ, sau đó kết hợp với thẻ DT, Danh từ NN, giới từ IN và thẻ VB - NN tạo thành cụm động từ, trong cùng một câu, trong đó DT là tùy chọn như trong Hình 3.10(f).

Ví dụ: The complaint page presents a form to register complaint against management of organization by simply clicking “add complaint button”.

Luật chuyển đổi: (\w+/VBZ(\w+/DT)?\w+/NN(\w+/IN |\w+/TO)\w+/VB\w+/NN)

3.2.3. Luật chuyển đổi cho Event

Có sáu quy tắc chuyển đổi cho các thành phần Event. Hình 3.11 trình bày các luật chuyển đổi cho Event (RE - Rule for Event), trong đó vòng tròn đỏ thể hiện tính năng bắt buộc phải có để áp dụng luật và vòng tròn màu xanh lá cây trình bày tính năng tùy chọn, có thể có hoặc không.



Hình 3.4. Các luật chuyển đổi cho thành phần Event.

RE1: Cụm động từ bao gồm dạng cơ sở VB với định thức DT hoặc tính từ JJ và Danh từ NN với giới từ IN ở đầu sẽ tương ứng với một Event như được minh họa trong hình 3.11(a).

Ví dụ: There is a provision to add a new movie through “add new” button which navigates to add movie page.

Luật chuyển đổi: ((\w+/IN|\w+/TO)\w+/VB (\w+/JJ|\w+/DT) \w+/NN)

RE2: Một cụm động từ bao gồm VBG với một tính từ JJ cộng với các thẻ NN tương ứng với một Event. Trong sơ đồ mô tả, hai vòng tròn màu xanh lá cây cho thấy phép lựa chọn giữa NN và JJ như thể hiện trong Hình 3.11(b).

Ví dụ: The user can enter required quantity and save it using save button.

Luật chuyển đổi: (\w+/VBG(\w+/JJ|\w+/NN) \w+/NN)

RE3: Nếu một động từ VB được theo sau bởi động từ quá khứ VBN và một danh từ NN hoặc nếu một động từ dạng cơ sở VB được theo sau bởi từ chỉ số lượng CD và NN, trong đó giới từ IN hoặc định thức DT có thể được trình bày tùy ý giữa cụm danh từ này, thì nó có thể biểu diễn một Event như trong Hình 3.11(c).

Ví dụ: The user can also select desired movie from the list of movies.

Luật chuyển đổi: (\w+/VB(\w+/VBN| (\w+/CD (\w+/IN)? (\w+/DT)?))\w+/NN)

RE4: Sự hiện diện của giới từ IN với một định thức tùy chọn DT theo sau bởi 2 danh từ NN sẽ tạo thành một Event như được thể hiện trong Hình 3.11(d).

Ví dụ: The page of shopping cart previews a list of selected products with a “checkout button”.

Luật chuyển đổi: (\w+/IN(\w+/DT)? \w+/NN \w+/NN)

RE5: Khi một động từ VBP được kết hợp với giới từ IN, theo sau bởi động từ cơ sở VB và danh từ NN, thì sự kết hợp này sẽ tạo thành một Event như hình 3.11(e).

Ví dụ: Once the user press “add to cart” button, a quantity window appears which presents a form to add quantity of product.

Luật chuyển đổi: Once the user press “add to cart” button, a quantity window appears which presents a form to add quantity of product.

RE6: Khi một động từ dạng VBG kết hợp với một cụm danh từ chứa định thức DT và một danh từ NN thì sẽ tương ứng với một Event như hình 3.11(f).

Ví dụ: On “selecting” a product, the detail of selected product appears on the screen.

Luật chuyển đổi: (\w+/VBG\w+/DT \w+/NN)

3.2.4. Luật chuyển đổi cho Action

Có hai luật chuyển đổi cho các Action. Điều quan trọng cần lưu ý là Action luôn được kích hoạt bởi một Event. Vì vậy, sự hiện diện của một Event tạo ra khả năng kích hoạt một Action. Hình 3.12 thể hiện các luật chuyển đổi cho Action (RA - Rule for Action), trong đó vòng tròn màu đỏ thể hiện tính năng bắt buộc phải có để thực thi luật và vòng tròn màu xanh lá cây trình bày tính năng tùy chọn, có thể có hoặc không.

Hình 3.5. Các luật chuyển đổi cho Action.

RA1: Một động từ dạng VBN được theo sau bởi một cụm động từ bao gồm sự kết hợp của thẻ VBG, NN hoặc JJ và thẻ NN sẽ tương ứng với một Action như trong Hình 3.12(a).

Ví dụ: After specifying the required shipping details, payment is executed using “pay” button.

Luật chuyển đổi: (\w+/VBN\w+/VBG (\w+/JJ|\w+/NN) \w+/NN)

RA2: Một động từ cơ sở VB được kết hợp với thẻ NN tùy chọn hoặc danh từ riêng PRP tùy chọn, cùng với một động từ VBG theo sau bởi hai thẻ NN liên tiếp, sự kết hợp này có thể biểu diễn một Action như trong hình 3.12(b).

Ví dụ: The user can enter required quantity and save it using save button.

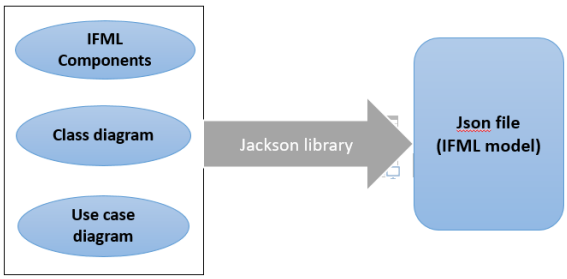
Luật chuyển đổi: (\w+/VB((\w+/NN|\w+/PRP))?(\w+/IN|\w+/VBG)(\w+/DT)?(\w+/NN) (\w+/NN)

3.3. Chuyển đổi đặc tả yêu cầu đầu vào thành mô hình IFML

Jackson là một thư viện dựa trên java rất phổ biến, hiệu quả và dễ sử dụng để tuần tự hóa hoặc ánh xạ các đối tượng java sang JSON và ngược lại. [11] Luận văn này sử dụng thư viện Jackson để tạo json file biểu diễn mô hình IFML từ các đặc tả đầu vào cần thiết.

Các đặc tả yêu cầu chức năng của giao diện người dùng bằng ngôn ngữ tự nhiên sau khi áp dụng các tập luật chuyển đổi trong mục 3.3, có thể sinh ra đặc tả các thành phần của mô hình IFML. Sau đó kết hợp đặc tả này với sơ đồ ca sử dụng và sơ đồ lớp, trích

xuất các giá trị cần thiết và sử dụng thư viện Jackson để tạo json file biểu diễn mô hình IFML.



Hình 3.6. Sử dụng Jackson library tạo json file biểu diễn mô hình IFML.

Với các tập luật chuyển đổi giới thiệu trên mục 3.3 phía trên, đặc tả yêu cầu đầu vào dạng ngôn ngữ tự nhiên sau khi áp dụng tập luật có thể sinh ra các thành phần của mô hình IFML và được lưu vào một file văn bản như ví dụ trong Hình 3.14.

```
View Container:
Book list
Book description page

View Component:
List of book
Details of selected book

Event:
Select
```

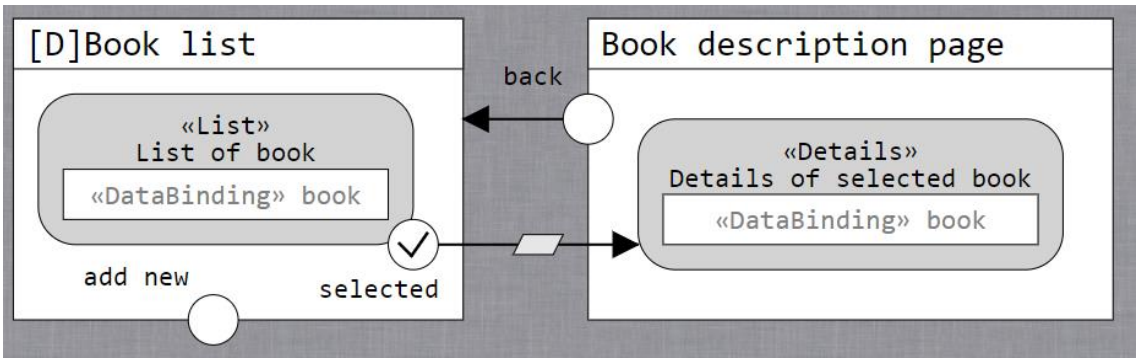
Hình 3.7. Các thành phần của mô hình IFML được sinh ra sau khi áp dụng luật chuyển đổi.

Một json file biểu diễn mô hình IFML bao gồm nhiều thành phần (element). Element này được tạo ra từ việc trích xuất các giá trị của văn bản chứa các thành phần IFML phía trên và sơ đồ lớp dạng văn bản. Hình 3.15 mô tả một element của json file biểu diễn mô hình IFML.


```
{
    "attributes": {
        "name": "Details of selected book",
        "stereotype": "details",
        "fields": [
            "title",
            "author"
        ],
        "collection": "book"
    },
    "metadata": {
        "graphics": {
            "position": {
                "x": 280,
                "y": 130
            },
            "size": {
                "height": 60,
                "width": 170
            }
        }
    },
    "id": "bookdetail",
    "type": "ifml.ViewComponent"
},
```


Hình 3.8. Một element của mô hình IFML trong json file.

Trong element phía trên, các giá trị *name*, *stereotype* của *attributes* và ifml *type* có thể được trích xuất từ đặc tả đầu vào mà mô tả các thành phần IFML. Các giá trị *fields* và *collection* của *attributes* sẽ được trích xuất từ sơ đồ lớp. Các giá trị metadata của graphic như size (width và height) được đặt giá trị mặc định là 60x170 đối với ViewComponent và 120x200 đối với ViewContainer. Sau khi sinh ra được các element mô tả mô hình IFML, tiếp tục kết hợp với sơ đồ ca sử dụng để tạo ra luồng liên kết giữa các element này và có được một json file hoàn chỉnh biểu diễn một mô hình IFML. Hình 3.16 mô tả một mô hình IFML do json file biểu diễn.

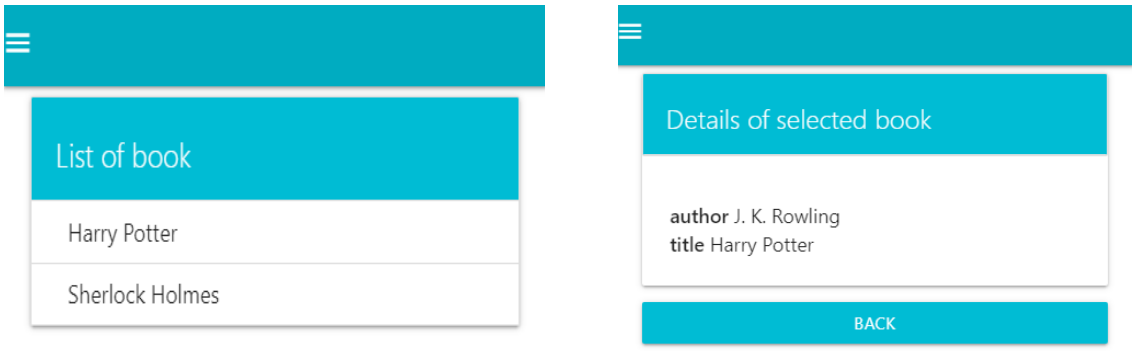


Hình 3.9. Mô hình IFML được sinh ra từ đặc tả yêu cầu.

3.4. Chuyển đổi mô hình IFML sang giao diện người dùng

Sau khi có được mô hình IFML mô tả giao diện người dùng, luận văn sử dụng công cụ mã nguồn mở IFMLEditor để chuyển đổi từ mô hình IFML thành bản mẫu giao diện người dùng. Bản mẫu giao diện này có thể biểu diễn các thành phần UI như ListView, EditText, TextView, Button, …

Hình 3.17 mô tả một bản mẫu giao diện cho danh sách book và thông tin book được sinh ra từ mô hình IFML trong hình 3.16 bằng công cụ IFMLEditor.



Hình 3.10. Bản mẫu giao diện người dùng được sinh ra từ mô hình IFML.

3.5. Tổng kết chương

Chương này đã trình bày về một quy trình hoàn chỉnh sinh tự động bản mẫu giao diện người dùng từ đặc tả yêu cầu chức. Luận văn áp dụng việc xử lý ngôn ngữ tự nhiên (OpenNLP) và các luật chuyển đổi để sinh ra đặc tả các thành phần mô hình IFML mô tả giao diện từ các đặc tả yêu cầu chức năng được viết bằng ngôn ngữ tự nhiên. Sau đó kết hợp đặc tả về thành phần IFML và các sơ đồ lớp, sơ đồ ca sử dụng để tạo ra jsonfile biểu diễn mô hình IFML thông qua thư viện Jackson. Cuối cùng là áp dụng công cụ IFML Editor để sinh ra bản mẫu giao diện người dùng mong muốn.

CHƯƠNG 4. CÀI ĐẶT VÀ THỰC NGHIỆM

Trong chương này, luận văn sẽ mô tả bộ công cụ và môi trường hỗ trợ để sinh tự động giao diện người dùng từ đặc tả yêu cầu chức năng. Sau đó, quy trình sinh tự động giao diện người dùng được áp dụng vào một nghiên cứu tình huống Quản lý sách nhằm đánh giá tính khả thi của phương pháp. Từ đó, luận văn sẽ đưa ra một số nhận xét về hiệu quả cùng các ưu nhược điểm của phương pháp.

4.1. Giới thiệu

Trong chương 3, luận văn đã nghiên cứu về quy trình sinh tự động giao diện người dùng từ đặc tả yêu cầu chức năng. Chương này trình bày cách cài đặt và sử dụng các công cụ, môi trường hỗ trợ nhằm áp dụng vào một tình huống nghiên cứu cụ thể là phần mềm Quản lý sách. Từ đó, luận văn đánh giá kết quả đạt được của thực nghiệm và tính khả thi của phương pháp, đồng thời cũng chỉ ra các hạn chế còn tồn tại và có thể mở rộng thêm trong tương lai.

4.2. Công cụ và môi trường hỗ trợ

Chương trình cài đặt phương pháp xây dựng bản mẫu giao diện người dùng từ đặc tả yêu cầu chức năng cho phần mềm Book Management được cài đặt bằng ngôn ngữ java và áp dụng các công cụ và thư viện hỗ trợ sau:

- Thư viện xử lý ngôn ngữ tự nhiên trong java OpenNLP được sử dụng để tách câu, mã hóa, gắn thẻ POS để có thể áp dụng các luật chuyển đổi cho thành phần IFML.
- Công cụ PlantTextUML giúp đặc tả các sơ đồ lớp, sơ đồ ca sử dụng dưới dạng văn bản nhằm sử dụng làm đầu vào cho việc sinh tự động mô hình IFML.
- Thự viện Java Jackson dùng để ánh xạ các đối tượng java sang json file, nhằm sinh tự động json file biểu diễn mô hình IFML từ các đặc tả yêu cầu.
- Công cụ mã nguồn mở giúp sinh tự động bản mẫu giao diện người dùng từ mô hình IFML tương ứng.

4.3. Nghiên cứu tình huống Quản lý sách

Các yêu cầu của chương trình phần mềm quản lý sách “Book Management” bao gồm các chi tiết của một ứng dụng cho phép quản lý thông tin liên quan đến sách.

Input: Đầu vào của chương trình bao gồm 3 file đặc tả dạng văn bản cho ứng dụng Book Management: UI_requirement.txt, Use_case.txt and Class_diagram.txt.

Output: Giao diện người dùng tương ứng cho ứng dụng Book Management.

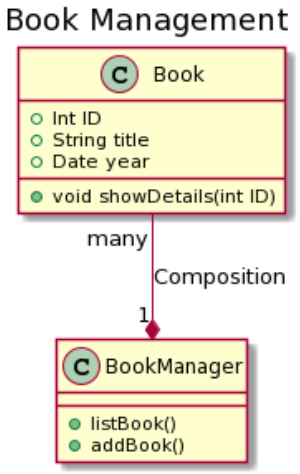
Bước 1: Xác định input, nội dung các đặc tả yêu cầu đầu vào trong input của phần mềm Book Management được mô tả như Hình 4.1 bên dưới.

> The main page exhibits a list of book. Particularly the title of each existing book is displayed on this. There is a provision to add a new book using add new button which navigates to add book page. It contains a form to add book detail the title of book and year of release. Once the form is successfully filled information can be saved using save button. In addition to show book feature the user can also select desired book from the list of book presented on main page in order to view detail. It directs to the book description page which provides detail of selected book information.

Hình 4.1. Đặc tả yêu cầu bằng ngôn ngữ tự nhiên (UI_requirement.txt).

Các Hình 4.2 và Hình 4.3 lần lượt mô tả các sơ đồ lớp và sơ đồ ca sử dụng cho phần mềm Book Management.

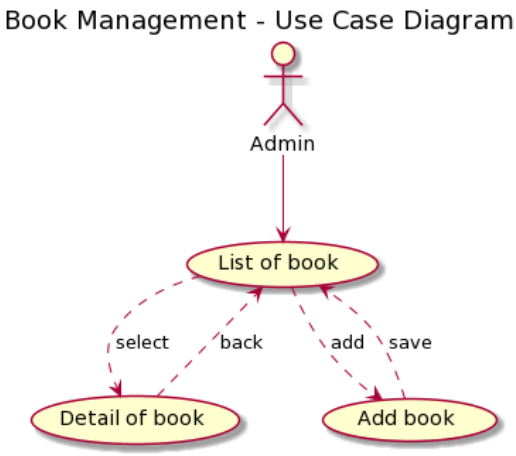
```
@startuml

title Book Management


class Book {
  +Int ID
  +String title
  +Date year
  +void showDetails(int ID)
}

class BookManager{
  +listBook()
  +addBook()
}

BookManager "1" *-up- "many" Book: Composition

@enduml
```



(a) (b)

Hình 4.2. Sơ đồ lớp dạng văn bản (Class_diagram.txt) (a) và dạng mô hình (b).

```
@startuml

title Book Management - Use Case Diagram

:Admin:
usecase (List of book)
usecase (Detail of book)
usecase (Add book)

Admin  --> (List of book)
(List of book) ..> (Detail of book): select
(List of book) ..> (Add book): add
(Add book) ..> (List of book): save
(Detail of book) ..> (List of book): back

@enduml
```



(a) (b)

Hình 4.3. Sơ đồ ca sử dụng dạng văn bản (Use_case.txt) (a) và dạng mô hình (b).

Bước 2: Với ba đặc tả đầu vào, chương trình cài đặt sẽ xử lý đặc tả bằng ngôn ngữ tự nhiên (UI_requirement.txt). Chương trình áp dụng thư viện xử lý ngôn ngữ tự nhiên OpenNLP, tách đoạn văn bản trong UI_requirement.txt thành các câu. Sau đó tách thành các từ đơn và xác định từ loại của câu.

VD: The main page exihibits a list of book.

⇨ DT + JJ + NN + VBZ + DT + NN + IN + NN

Sau khi xác định được từ loại của các từ trong câu, chương trình sẽ cài đặt thuật toán kiểm tra xem câu đó có thỏa mãn các tập luật sinh IFML đã giới thiệu trong mục 3.3 không. Nếu câu đã cho chứa các tập luật chuyển đổi như một tập con thì nó sẽ xác định được một thành phần IFML. Với câu mô tả trên ví dụ, nó thỏa mãn luật chuyển đổi cho View Container (JJ + NN + VBZ). Tương tự, kiểm tra cho tất cả các câu trong đặc tả dạng văn bản trong UI_requirement.txt, chương trình sẽ xác định được các thành phần IFML theo các tập luật như sau. Tuy nhiên, tập luật chỉ có thể sinh ra các thành phần View Container, View Component, Event và Action, nhưng lại chưa thể phân biệt được các loại View Componnent khác nhau. Luận văn hiện đang sử dụng một tập các keyword mặc định của các thành phần ViewConponent như List, Form, Details, Hierarchy, Scroller, tìm kiếm các keyword trong câu văn bản đặc tả đáp ứng tập luật sinh View Component để xác định loại View Component tương ứng.

Bảng 4.1. Các thành phần mô hình IFML cho ứng dụng Book Management

No	Requirement	Transformation rule	View Container	View Component	Event	Action
1	The main page exhibits a list of book.	1. JJ + NN + VBZ 2. VBZ + DT + NN + IN + NN	Main page	List of book		
2	Particularly the title of each existing book is displayed on this.					
3	There is a provision to add a new book using add new button which navigates to add new book.	1. JJ + NN + VBZ 2. IN + DT + NN + NN	Add book page		Add	
4	It contains a form to add book detail the title of book and year of release.	1. VBZ + DT + IN + NN + NN		Form to add book		
5	Once the form is successfully filled information can be saved using save button.	1. VBP + IN + VB + NN 2. VB + NN + VBG + NN + NN			Save	Saved
6	In addition to show book feature the user can also select desired book from the list of book presented on main page in order to view its detail.	1. VBN + IN + JJ + NN 2. NN + IN + NN + VBN 3. IN + VB + JJ + NN	Main page	List of book	Select	
7	It directs to the book description page which provides detail of selected book information.	1. NN + NN + NN + WDT + VBZ 2. VBZ + DT + NN + IN + NN	Book description page	Details of selected book		

Các thành phần của mô hình IMFL sau khi sinh ra sẽ được lưu vào file đặc tả thành phần IFML dạng văn bản IFML_Component.txt như Hình 4.4.

```
View Container, Main page
View Component, List of book
View Container, Add book page
Event, Add
View Component, Form to add book
Event, Save
Action, Saved
View Container, Main page
View Component, List of book
Event, select
View Container, Book description page
View Conponent, Detail of selected book
```

Hình 4.4. Thành phần IFML cho phần mềm Book Management được lưu trong IFML_Component.txt.

Mô hình IFML cho ứng dụng Book Management bao gồm ba View Container (Main page, Add movie page, Movie description page), ba View Component (List of movie, Form to add movie, Detail of selectec movie), ba Event (Select, Add, Save) và một Action (Saved).

Bước 3: Sau khi sinh ra được các thành phần IFML, kết hợp 3 đặc tả IFML_Component.txt, Use_case.txt và Class_diagram.txt để sinh ra json file biểu diễn mô hình IFML thông qua thư viện Jackson trong java. Chương trình đọc các dữ liệu cần thiết cho mô hình IFML từ ba file đặc tả trên và lưu chúng vào các đối tượng java, sau đó sử dụng thư viện Jackson để ánh xạ các đối tượng java sang các thành phần tạo thành json file. Json file bao gồm 2 thành phần chính là elements và relations. Elements thể hiện các thành phần của mô hình IFML và relations thể hiện mối quan hệ của chúng. Hình 4.5 và Hình 4.6 mô tả ví dụ về elements và relations trong json file cho phần mềm Book Management.


```
{
    "attributes": {
        "name": "Form to add book",
        "stereotype": "form",
        "fields": [
            "title",
            "year"
        ]
    },
    "metadata": {
        "graphics": {
            "position": {
                "x": 50,
                "y": 370
            },
            "size": {
                "height": 60,
                "width": 150
            }
        }
    },
    "id": "formtoaddbook",
    "type": "ifml.ViewComponent"
},
```


Hình 4.5. Element cho thành phần IFML "Form to add book".

Các giá trị của attribute name “Form to add book”, stereotype “form”, type “ifml.ViewComponent” được lấy từ IFML_Component.txt, id “formtoaddbook” được lấy từ name và bỏ các dấu cách. Các giá trị fields của attributes “title” và “year” được lấy từ Class_diagram.txt. Các giá trị graphic trong metadata là các giá trị cho vị trí của thành phần View Container được hard code các giá trị mặc định, và các giá trị của View Component được set sao cho size và position luôn nhỏ hơn View Container chứa nó. Sau khi sinh ra được các element biểu diễn các thành phần của mô hình IFML, chương trình tiếp tục sinh ra các relations thể hiện mối quan hệ giữa các elements này nhờ lấy các giá trị từ Use_case.txt.

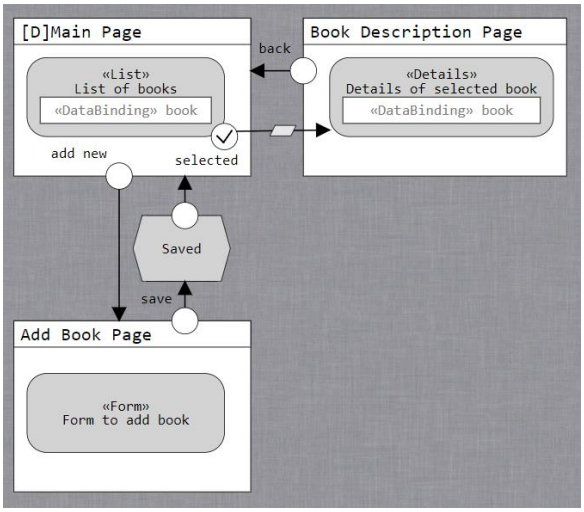

```
{
    "type": "hierarchy",
    "parent": "mainpage",
    "child": "listofbook"
},
{
    "type": "hierarchy",
    "parent": "mainpage",
    "child": "add"
},
{
    "type": "source",
    "flow": "addflow",
    "source": "add"
},
{
    "type": "target",
    "flow": "addflow",
    "target": "formtoaddbook"
},
```


Hình 4.6. Ví dụ về các relations trong json file của mô hình IFML cho phần mềm Book Management.

Hình 4.6. trình bày một vài ví dụ về relations giữa các element trong mô hình IFML. Main page chứa các thành phần con là View Component “listofbook” và Event “add”. Từ source là event “Add” của mainpage có thể liên kết tới target là View Component “formtoaddmovie”.

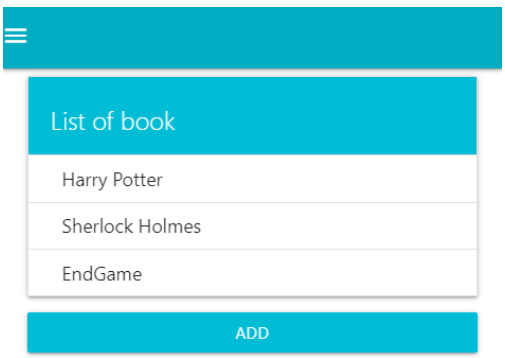
Bước 4: Sau khi sinh ra json file, sử dụng công cụ hỗ trợ IFML Editor để sinh ra bản mẫu giao diện người dùng.

Json file được sinh ra trong chương trình java cho phần mềm Book Management khi mở bằng công cụ IFML Editor sẽ biểu diễn được một mô hình IFML như Hình 4.7.

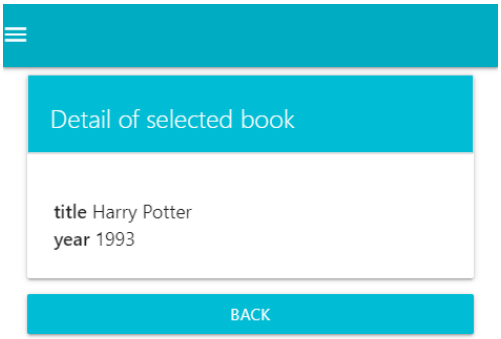


Hình 4.7. Mô hình IFML cho phần mềm Book Management.

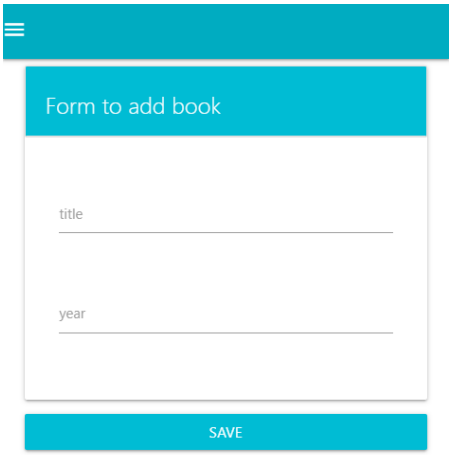
Cuối cùng sử dụng công cụ mã nguồn mở IFML Editor để chuyển đổi json file cho mô hình IFML trên sang bản mẫu giao diện người dùng tương ứng như trong Hình



(a)



(b)



(c)

Hình 4.8. Bản mẫu giao diện cho ứng dụng Book Management.

Hình 4.8 trình bày bản mẫu giao diện người dùng cho ứng dụng Book Management. Bản mẫu giao diện này đáp ứng được các yêu cầu trong đặc tả ban đầu UI_requirement của người dùng. Bản mẫu giao diện gồm ba trang giao diện chính, trang đầu tiên cũng là trang mặc định nhằm hiển thị danh sách các book theo title (hình 4.6a). Khi người dùng click vào một book, Event select được gọi và chuyển sang trang giao diện mô tả chi tiết về book được chọn bao gồm title và year (hình 4.6b). Và giao diện trong hình 4.6(c) nhằm mục đích giúp người dùng điền thông tin cho một book mới, khi người dùng ấn save và lưu nó vào danh sách, chuyển về giao diện mặc định ban đầu. Khi có thêm thay đổi về yêu cầu của ứng dụng Book Management, chỉ cần cập nhật lại đặc tả yêu cầu, giao diện sẽ được sinh tự động lại theo yêu cầu mới mà người phát triển không cần tự bổ sung mã nguồn thủ công.

4.4. Kết quả thực nghiệm

Áp dụng phương pháp sinh tự động giao diện người dùng từ đặc tả yêu cầu ban đầu, thực nghiệm trên phần mền Quản lý sách, đã sinh ra được bản mẫu giao diện đáp ứng với các yêu cầu của người dùng được mô tả trong phần đặc tả yêu cầu chức năng bằng ngôn ngữ tự nhiên trong mục 4.3. Tuy nhiên, trong bối cảnh này, có thể nói rằng quy mô của nghiên cứu tình huống là hơi nhỏ để thực hiện đánh giá thực tế. Ở đây, mục tiêu là chứng minh tính khả thi của phương pháp đề xuất và nó đạt được bằng cách thông qua một nghiên cứu tình huống nhất định.

4.5. Đánh giá và thảo luận

Ưu điểm của phương pháp đề xuất là có thể sử dụng các yêu cầu ngôn ngữ tự nhiên không bị giới hạn mà không cần bất kỳ mẫu cụ thể nào. Tuy nhiên, ngôn ngữ tiếng Anh có thể được viết theo nhiều kiểu khác nhau và không thể quản lý từng kiểu trong phương pháp đề xuất. Về vấn đề này, trong mục 2.4.1 luận văn đã đưa ra một vài hướng dẫn để viết các đặc tả yêu cầu văn bản trên cơ sở tài liệu [12][13]. Các hướng dẫn này là quy trình chuẩn thường được sử dụng để đặc tả các yêu cầu phần mềm. Tuân theo các hướng dẫn đã cho để viết các yêu cầu văn bản sẽ cải thiện độ chính xác của phương pháp sinh tự động giao diện. Tuy phương pháp đề xuất có thể tạo các cấu trúc IFML từ các yêu cầu văn bản có thể được viết theo bất kỳ kiểu nào nhưng độ chính xác của nó có thể bị ảnh hưởng tùy thuộc vào kiểu của các yêu cầu văn bản nhất định.

Hai lợi ích chính của phương pháp đề xuất là:

- Nó cho phép xác nhận các yêu cầu ban đầu trong giai đoạn đầu của SDLC. Đặc biệt, nó ngay lập tức tạo ra các mô hình IFML mục tiêu và giao diện từ các yêu cầu văn bản ban đầu để trực quan hóa hành vi và tương tác của các giao diện người dùng. Điều này dẫn đến việc xác nhận các yêu cầu ban đầu trong thời gian

ngắn vì việc phát triển thủ công các mô hình IFML và bản mẫu giao diện từ các yêu cầu ban đầu là một hoạt động tốn thời gian.

- Nó làm giảm đáng kể thời gian phát triển hệ thống vì các mô hình IFML và bản mẫu giao diện được tạo tự động có thể được sử dụng trong các giai đoạn SDLC tiếp theo.

Tuy đạt được một vài kết quả trên, phương pháp đề xuất vẫn còn có các hạn chế và có khả năng mở rộng cao và hỗ trợ các cải tiến hơn nữa. Ví dụ, có thể kết hợp nhiều tập luật hơn trong phương pháp được đề xuất để tạo các cấu trúc IFML từ đặc tả yêu cầu chức năng bằng ngôn ngữ tự nhiên mà không cần kết hợp với sơ đồ lớp và sơ đồ ca sử dụng dạng văn bản. Như với đặc tả trong ví dụ về phần mềm quản lý sách ở mục 4.3, với câu văn thứ 4 trong đặc tả "*It contains a form to add book detail the title of book and year of release*.", có thể nghiên cứu thêm và mở rộng các phương pháp xử lý ngôn ngữ tự nhiên, các quy tắc mới để sinh ra được lớp Book và các trường của lớp Book như Tittle, Year mà không cần kết hợp với sơ đồ lớp như trongg phương pháp đề xuất của luận văn.

Trong trường hợp này này, luận văn chỉ cung cấp ý tưởng nền tảng của phương pháp được đề xuất bằng cách nhắm mục tiêu bốn cấu trúc IFML quan trọng nhất, tức là View Container, View Component, Actions và Events. Việc sinh tự động thành phần View Component của mô hình IFML được đưa ra nhưng việc phân loại chi tiết hơn các loại thành phần IFML như List, Detail, Form, Hierarchy, Scroller hiện vẫn đang bị thiếu và cần sử dụng bộ keyword có sẵn để xác định. Các thành phần chi tiết của IFML còn thiếu như vậy có thể được nghiên cứu và phát triển thêm trong tương lai.

4.6. Tổng kết chương

Trong chương này, luận văn trình bày cách cài đặt và sử dụng các công cụ, môi trường hỗ trợ nhằm áp dụng vào một tình huống nghiên cứu cụ thể là phần mềm Quản lý sách để sinh tự động giao diện người dùng. Luận văn cũng đã đưa ra các đánh giá về kết quả đạt được, bản mẫu giao diện được sinh ra cho phần mềm Quản lý sách đã đáp ứng được các yêu cầu cơ bản được mô tả trong đặc tả yêu cầu văn bản đầu vào của người dùng. Tuy quy mô nghiên cứu còn hơi nhỏ và chưa thể thực hiện đánh giá thực tế nhưng có thể đánh giá được tính khả thi của phương pháp đề xuất. Cuối cùng, luận văn thảo luận về các ưu điểm của phương pháp và chỉ ra những điểm hạn chế có thể mở rộng trong tương lai.

CHƯƠNG 5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Trong các ứng dụng ngày nay, giao diện người dùng đóng vai trò rất quan trọng và nó ngày càng đa dạng và phức tạp. Vì vậy, việc phát triển giao diện người dùng trong giai đoạn đầu của vòng đời phát triển phần mềm vô cùng mất thời gian và tốn kém công sức, nguồn lực. Người ta ước tính rằng việc phát triển một ứng dụng yêu cầu 48% code được áp dụng trên các giao diện người dùng và 50% thời gian phát triển được dành cho việc biểu diễn các giao diện người dùng. Điều này dẫn đến nhu cầu về việc phát triển một quy trình tự động sinh bản mẫu giao diện người dùng từ đặc tả yêu cầu bằng cách áp dụng kỹ nghệ hướng mô hình (MDE).

IFML là một ngôn ngữ mô hình hóa độc lập nền tảng dựa trên nguyên tắc MDE cung cấp mô hình hóa các giao diện người dùng tinh vi cho các ứng dụng khác nhau như di động, web... IFML bao gồm nhiều tính năng phù hợp để phát triển các mô hình/bản mẫu giao diện người dùng ban đầu từ các đặc tả yêu cầu. Hơn nữa, nó cũng giảm thời gian phát triển vì các mô hình IFML đã tạo có thể được sử dụng trong các giai đoạn SDLC tiếp theo bằng cách áp dụng các phép chuyển đổi mô hình.

Do đó, luận văn này cung cấp một phương pháp sinh tự động bản mẫu giao diện người dùng từ đặc tả yêu cầu chức năng thông qua mô hình hóa luồng tương tác IFML. Quy trình này giúp tách biệt các phức tạp của lập trình khỏi nền tảng thực thi, và tăng hiệu suất phát triển nhờ việc tái sử dụng thiết kế hệ thống.

5.1. Kết quả đạt được

Sau quá trình tìm hiểu và nghiên cứu, luận văn đã đạt được các kết quả đóng góp như sau:

- Luận văn đã nghiên cứu tìm hiểu về các kỹ thuật MDE, mô hình IFML, các đặc tả yêu cầu chức năng dạng văn bản, nghiên cứu áp dụng các thư viện xử lý ngôn ngữ tự nhiên openNPL, thư viện sinh json file Jackson và công cụ hỗ trợ IFML Editor để đưa ra được ý tưởng về một quy trình hoàn chỉnh nhằm sinh tự động bản mẫu giao diện người dùng từ các đặc tả yêu cầu chức năng ban đầu.
- Luận văn cũng đã áp dụng phương pháp xử lý ngôn ngữ tự nhiên, cài đặt chương trình để kiểm tra các câu trong văn bản có chứa các luật chuyển đổi sinh thành phần IFML không và từ đó sinh thành phần IFML tương ứng.
- Luận văn đã đưa ra phương pháp kết hợp thành phần IFML với các đặc tả sơ đồ lớp, sơ đồ ca sử dụng và áp dụng thư viện Java Jackson để cài đặt chương trình sinh tự động json file mô tả mô hình IFML từ các đặc tả.

- Cuối cùng, luận văn đã áp dụng phương pháp sinh tự động bản mẫu giao diện người dùng vào bài toán thực tế Book Management để đánh giá tính khả thi của phương pháp.

5.2. Hướng phát triển

Tuy đã đạt được các kết quả sơ bộ về việc sinh tự động bản mẫu giao diện người dùng từ đặc tả yêu cầu chức năng, phương pháp trình bày trong luận văn vẫn còn các vấn đề hạn chế. Do đó luận văn đề xuất các hướng phát triển tiếp theo trong tương lai:

- Phương pháp sinh bản mẫu giao diện từ ba loại đặc tả yêu cầu đầu vào là đặc tả yêu cầu chức năng bằng ngôn ngữ tự nhiên, sơ đồ lớp và sơ đồ ca sử dụng. Do đó cần sự tham gia của người dùng và người phân tích thiết kế trong giai đoạn đầu để cung cấp các đặc tả yêu cầu. Trong tương lai, phương pháp có thể bổ sung thêm các quy tắc xử lý ngôn ngữ tự nhiên, các tập luật chuyển đổi mới để có thể sinh tự động giao diện chỉ từ các đặc tả yêu cầu chức năng bằng ngôn ngữ tự nhiên của người dùng, mà không cần đến các sơ đồ lớp và sơ đồ ca sử dụng.
- Luận văn mới chỉ áp dụng phương pháp trong các bài toán nhỏ đơn giản, chưa thử nghiệm với các chương trình đa dạng và phức tạp nên chưa đánh giá thực tế được hết phạm vi các bài toán mà phương pháp có thể áp dụng. Trong tương lai, luận văn có thể xây dựng một bộ đầu vào các đặc tả cho nhiều bài toán đa dạng khác nhau để áp dụng quy trình sinh tự động bản mẫu giao diện nhằm đưa ra các số liệu thống kê, đánh giá được phạm vi áp dụng và độ chính xác của phương pháp.

TÀI LIỆU THAM KHẢO

[1] M. D. Lozano, P. Gonzalez and I. Ramos, "User interface specification and modelling in an object-oriented environment for automatic software development", *in Proc. Int. Conf. on Technology of Object-Oriented Languages and Systems (TOOLS 34)*, 2000.

[2] C. He and G. Mussbacher, "Model-Driven Engineering and Elicitation Techniques: A Systematic Literature Review", *in Proc. REW*. 2016.

[3] M. Brambilla and P. Fraternali, "Large-scale Model-Driven Engineering of web user interaction: The WebML and WebRatio experience", *Science of Computer Programming*, 2014.

[4] Object Management Group IFML Specifications Version 1, [Accessed online: 19-Nov-2019]. http://www.omg.org/spec/IFML/1.0/

[5] N. Laaz and S. Mbarki, "Integrating IFML models and owl ontologies to derive UIs web-Apps", *in Proc. Of International Conference on Information Technology for Organizations Development (IT4OD)*, 2016.

[6] D. Liu, K. Subramaniam, A. Eberlein, and B. H. Far, "Natural language requirements analysis and class model generation using UCDA". *In Proc. of International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems,* Berlin, Heidelberg, 2004.

[7] M. Brambilla, J. Cabot, and M. Wimmer, "*Model-Driven Software Engineering in Practice: Second Edition*", Morgan & Claypool Publishers, 2017.

[8] R. S. Wazlawick, "*Oject-Oriented Analysis and Design for Information Systems, Modeling with UML, OCL, and IFML*", Elsevier Inc Pulishers, 2014.

[9] Natural Language Process OpenNLP, The Apache Software Foundation pulisher, http://opennlp.apache.org/.

[10] M. Hamdani, W.H. Butt, M. W. Anwar, I. Ahsan, F. Azam and M. A. Ahmed, "A Novel Framework to Automatically Generate IFML Models from Plain Text Requirements", 2017

[11] Java json library: Jackson library, https://www.tutorialspoint.com/jackson/index.htm

[12] S. S. Larissa da Costa, V. V. G. Neto and J. Lopes de Oliveira, "A User Interface Stereotype to build Web Portals," *in Proc. Of 9th Latin American Web Congress*, Ouro Preto, Brazil, Oct. 2014

[13] M. A. Ahmed, W.H. Butt, I. Ahsan, M. W. Anwar, M. Latif and F. Azam, "A Novel Natural Language Processing (NLP) Approach to Automatically Generate Conceptual

Class Model from Initial Software Requirements" *In Proc. of 8th International Conference on Information Science and Applications (ICISA)*, 2017.

[14] J. Bézivin. On the unification power of models. *Software and System Modelling*, 4(2):171–188, 2005.

[15] M. Brambilla and P. Fraternali, "*Interaction Flow Modeling Language: Model-Driven UI Engineering of Web and Mobile Apps with IFML*", Morgan Kaufmann Publisher, 2014