BỘ GIÁO DỤC VÀ ĐÀO TẠO
**ĐẠI HỌC ĐÀ NẴNG**

**NGUYỄN VĂN ĐỊNH**

# NGHIÊN CỨU VÀ ỨNG DỤNG KIỂM CHỨNG MÔ HÌNH CHO CÁC HỆ THỐNG PHÁT TRIỂN TRÊN MÔI TRƯỜNG LUSTRE/SCADE

**Chuyên ngành**: **Khoa học máy tính**
**Mã số:** **60.48.01**

**TÓM TẮT LUẬN VĂN THẠC SĨ KỸ THUẬT**

**Đà Nẵng - Năm 2013**

Công trình được hoàn thành tại

**ĐẠI HỌC ĐÀ NẴNG**

**Người hướng dẫn khoa học: TS. NGUYỄN THANH BÌNH**

**Phản biện 1**: **TS. PHẠM MINH TUẤN**

**Phản biện 2**: **TS. NGUYỄN QUANG THANH**

Luận văn được bảo vệ trước Hội đồng chấm Luận văn tốt nghiệp thạc sĩ Kỹ thuật họp tại Đại học Đà Nẵng vào ngày 16 tháng 11 năm 2013.

*Có thể tìm hiểu luận văn tại:*

- Trung tâm Thông tin - Học liệu, Đại Học Đà Nẵng

# MỞ ĐẦU

## 1. Tính cấp thiết của đề tài

Các lĩnh vực công nghệ thông tin và truyền thông bao gồm cả các hệ thống phần cứng và phần mềm trong thời đại ngày nay thực sự phát triển mạnh mẽ. Phạm vi, kích thước và độ phức tạp của các phần mềm đòi hỏi sự an toàn, độ tin cậy ở mức cao nhất trong nhiều lĩnh vực (thương mại, khí tài trong lực lượng vũ trang và năng lượng). Hiện tại những phương pháp xác minh sẽ không thể đáp ứng để đảm bảo tính hiệu quả, độ tin cậy với những phần mềm thế hệ tiếp theo này. Các qui trình kiểm định mới đã được phát triển làm tăng khả năng kiểm thử hệ thống phần mềm, với các kỹ thuật phân tích như các phương thức mô hình hóa. Những qui trình này đảm bảo được các chức năng nâng cao cần thiết trong các hệ thống phần mềm hiện đại (như hệ thống phần mềm điều khiển trên máy bay Air Bus, hệ thống giám sát và điều khiển tàu điện ngầm, các hệ thống quản lý năng lượng nhà máy điện hạt nhân,…) như giá thành, về sự an toàn chắc chắn trong hệ thống phần mềm.

Việc đảm bảo chất lượng phần mềm là một trong những công đoạn khó khăn nhất của việc phát triển phần mềm. Trong đó, việc đảm bảo tính đúng đắn của bản thiết kế ở bước sớm nhất có thể là một thách thức lớn nhất đối với bất kì quy trình phát triển phần mềm nào. Từ trước đến nay, phương pháp giả lập và kiểm thử thường được sử dụng để kiểm tra các bản thiết kế. Tuy nhiên phương pháp này bộc lộ nhiều khiếm khuyết, trong đó điểm yếu nghiêm trọng nhất chính là không thể khẳng định được chương trình đã hết lỗi hoặc ước lượng được số lỗi có thể sót lại trong bản thiết kế.

Kiểm chứng mô hình là một kỹ thuật kiểm chứng tự động các hệ thống hữu hạn trạng thái. Kiểm chứng mô hình xác minh tính đúng đắn của một mô hình bằng việc xác định xem các thuộc tính người dùng mong muốn có được thỏa mãn bởi mô hình đó hay không. Về nguyên tắc hoạt động, hệ thống cần kiểm chứng sẽ được mô hình hóa. Công cụ kiểm chứng sẽ kiểm tra mô hình có thỏa mãn các thuộc tính được cho hay không. Nhờ khả năng duyệt qua tất cả các trạng thái trong mô hình mà tính đúng đắn của kết quả kiểm chứng mô hình luôn được đảm bảo.

Kiểm chứng mô hình đề cập trong đề tài nghiên cứu cho các hệ thống phát triển trên môi trường Lustre/SCADE, trong đó Lustre là ngôn ngữ lập trình đồng bộ cho hệ thống phản ứng. SCADE là môi trường để xây dựng, phát triển các phần mềm. SCADE bao gồm bộ công cụ phần mềm hữu hiệu cho phép các kỹ sư hệ thống tạo ra mô hình, mô tả hệ thống phần mềm trước trong vòng đơi phát triển, cho phép phân tích các hành vi yêu cầu và sau đó dùng để tự động sinh mã và các ca kiểm thử trong hệ thống phần mềm. Điểm nhấn mạnh của việc phát triển dựa trên mô hình này là tập trung vào việc mô hình hóa, mô phỏng và phân tích, tự động sinh mã và các ca kiểm thử. Điều này giảm chi phí phát triển sản phẩm bởi vì thứ nhất là tìm ra những yếu điểm trước trong vòng đời phát triển, tránh thực hiện lại các công việc trong trường hợp lỗi xảy ra ở giai đoạn kiểm thử tích hợp. Thứ hai tự động sinh mã và các ca kiểm thử phần mềm. Phát triển dựa trên mô hình trong môi trường SCADE có ý nghĩa đặc biệt làm giảm chi phí phát triển và tăng chất lượng sản phẩm phần mềm.

Kiểm chứng mô hình cho các hệ thông phát triển trên Lustre/SCADE cung cấp phương pháp luận để làm tăng độ tin cậy, giảm thiểu thời gian phát triển, các lỗi xảy ra trên các hệ thống phần mềm. Kiểm chứng mô hình thỏa mãn các điều kiện cần có với một công cụ tự động như sau:

- ✓ Có cơ sở hình thức để xây dựng được các công cụ có tính thực thi.
- ✓ Có khả năng liên kết giữa các giai đoạn trong vòng đời phần mềm giúp cho việc dễ dàng tích hợp giữa các pha trong vòng đời phần mềm.
- ✓ Tính ổn định cao, nhất là với những phần mềm phức tạp.
- ✓ Có khả năng phát hiện lỗi và sửa lỗi.

Với lợi ích to lớn của kiểm chứng mô hình đặc biệt là kiểm chứng mô hình trên các phần mềm hiện đại đòi hỏi độ an toàn và tin cậy mức cao, đây trở thành một vấn đề nóng được rất nhiều nhà khoa

học, chuyên gia trong nước, trên thế giới quan tâm. Với lý do trên và được sự đồng ý của cán bộ hướng dẫn TS. Nguyễn Thanh Bình, tôi chọn hướng nghiên cứu luận văn này, với đề tài: ***"Nghiên cứu và ứng dụng kiểm chứng mô hình cho các hệ thống phát triển trên môi trường Lustre/SCADE"***.

## 2. Mục đích nghiên cứu

Nội dung của đề tài là nghiên cứu kiểm chứng mô hình cho các hệ thống phát triển trên môi trường Lustre/SCADE nhằm tăng độ tin cậy, tính an toàn cho các hệ thống, giảm thời gian phát triển và lỗi xảy ra trong hệ thống phần mềm. Bên cạnh đó, đề tài cung cấp phương pháp luận tốt nhất để thực hiện việc kiểm định hệ thống phần mềm. Để giải quyết, đạt được những yêu cầu đó trong đề tài cần tìm hiểu cơ sở lý thuyết về kiểm chứng mô hình như các phương pháp và qui trình thực hiện kiểm chứng mô hình trên các hệ thống phản ứng. Nghiên cứu môi trường Lustre/SCADE, tìm hiểu các công cụ kiểm chứng mô hình phổ biến hiện nay.

Trên cơ sở nghiên cứu, tìm hiểu các vấn đề lý thuyết nêu trên trong đề tài sử dụng ngôn ngữ Lustre viết chương trình và dùng công cụ kiểm chứng mô hình Lesar để kiểm chứng các hệ thống phản ứng, thông qua số ứng dụng cụ thể.

## 3. Đối tượng và phạm vi nghiên cứu

Đối tượng nghiên cứu:

- ✓ Nghiên cứu cơ sở lý thuyết về kiểm chứng mô hình.
- ✓ Nghiên cứu ngôn ngữ đồng bộ Lustre/SCADE.
- ✓ Nghiên cứu hệ thống phản ứng.
- ✓ Các công cụ kiểm chứng mô hình (Lesar, NuSMV,SPIN)
- ✓ Nghiên cứu bộ SCADE Suite và ứng dụng trong thực tế.
- ✓ Phương pháp và quy trình kiểm chứng mô hình cho các hệ thống phản ứng, mô hình hóa hệ thống phản ứng với Lustre và xác minh bằng công cụ Lesar.

Phạm vi nghiên cứu:

- ✓ Cơ sở lý thuyết về kiểm chứng mô hình phần mềm cho các hệ thống phản ứng.

- ✓ Tìm hiểu ngôn ngữ lập trình đồng bộ luồng dữ liệu Lustre để viết chương trình.
- ✓ Giới thiệu các công cụ kiểm chứng, phân tích để lựa chọn công cụ kiểm chứng Lesar để xác minh mô hình.
- ✓ Giới thiệu về môi trường SCADE và bộ phần mềm SCADE Suite® để nêu lên được vai trò và ứng dụng về môi trường này.
- ✓ Thử nghiệm kiểm chứng mô hình cho hệ thống phản ứng.

## 4. Phương pháp nghiên cứu

Trên cơ sở nội dung nghiên cứu nêu trên, tác giả tiến hành nghiên cứu theo phương pháp:

- ✓ Đọc tài liệu, lựa chọn nội dung phù hợp từ tài liệu tham khảo và nhiều bài báo khoa học nước ngoài có liên quan mật thiết về đề tài.
- ✓ Phân tích, xác định trọng tâm và xây dựng cơ sở lý luận về vấn đề nghiên cứu và trên cơ sở đó thực nghiệm giải quyết vấn đề thông qua việc thử nghiệm cụ thể.
- ✓ Đánh giá và rút ra bài học kinh nghiệm thực tiễn.

## 5. Ý nghĩa khoa học và thực tiễn của đề tài

Với những nội dung nghiên cứu đã được nêu trên, đề tài mang lại ý nghĩa thiết thực về khoa học và thực tiễn.

Ý nghĩa khoa học:

Nghiên cứu kiểm chứng mô hình phần mềm cung cấp phương pháp luận để tăng độ tin cậy, giảm thiểu lỗi xảy ra ở mức chấp nhận trong việc thiết kế và xây dựng phần mềm nói chung và phần mềm hệ thống phản ứng nói riêng.

Ý nghĩa thực tiễn:

- ✓ Kiểm chứng mô hình là một kỹ thuật kiểm chứng tự động các hệ thống hữu hạn trạng thái.
- ✓ Ứng dụng kiểm chứng mô hình thử nghiệm cho các hệ thống phản ứng hiện đại trên môi trường Lustre/SCADE nhằm giảm thời gian, chi phí phát triển, kiểm thử tự động giúp phát

hiện lỗi sớm ở mức thiết kế cho hệ thống phần mềm, phần cứng.

## 6. Cấu trúc của luận văn

Cấu trúc của luận văn bao gồm các chương mục chính như sau:

PHẦN MỞ ĐẦU

Nêu lên tính cấp thiết đề tài, mục đích nghiên cứu, đối tượng và phạm vi nghiên cứu, phương pháp nghiên cứu, ý nghĩa khoa học và tính thực tiễn của đề tài.

CHƯƠNG 1- TỔNG QUAN KIỂM CHỨNG MÔ HÌNH

Trong chương này, chúng tôi trình bày những cơ sở lý thuyết có liên quan: kiểm chứng mô hình. Qui trình và phương pháp kiểm chứng kiểm chứng mô hình. Ưu điểm và nhược điểm của kiểm chứng mô hình.

CHƯƠNG 2- CÁC HỆ THỐNG PHẢN ỨNG VÀ MÔI TRƯỜNG LUSTRE/SCADE

Trong chương này, chúng tôi tập trung trình bày về hệ thống phản ứng, ngôn ngữ Lustre, môi trường SCADE và dòng sản phẩm SCADE Suite®.

CHƯƠNG 3- ỨNG DỤNG KIỂM CHỨNG MÔ HÌNH CHO CÁC HỆ THỐNG PHÁT TRÊN MÔI TRƯỜNG LUSTRE/SCADE

Trong chương này, chúng tôi tập trung phân tích và lựa chọn công cụ kiểm chứng mô hình. Xây dựng qui trình kiểm chứng mô hình trên Lesar và thử nghiệm kiểm chứng mô hình cho hệ thống phản ứng thông qua hai ứng dụng.

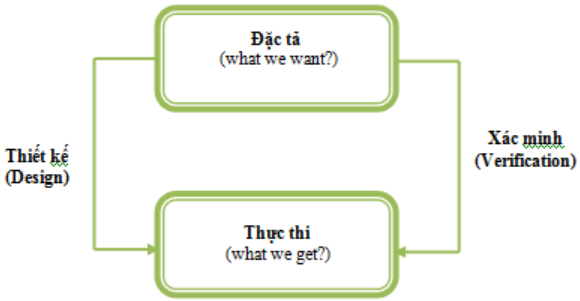
KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Nêu lên những kết quả nghiên cứu đạt được, những điểm hạn chế trong đề tài và hướng phát triển của luận văn.

# CHƯƠNG 1 -
# TỔNG QUAN KIỂM CHỨNG MÔ HÌNH

## 1.1. GIỚI THIỆU

Kiểm chứng mô hình phần mềm đã có lịch sử phát triển từ khá sớm với mục đích đạt được là phải tự động hoá quá trình xác minh các hệ thống, cho đến nay đã phát triển lên thành nhiều phương pháp luận [1]. Từ khi bắt đầu phát triển theo hướng này, người ta đã xác định được điều kiện tiên quyết của tự động hoá quá trình xác minh gồm hai yếu tố: ngữ nghĩa hình thức và ngôn ngữ đặc tả.




***Hình 1-1: Mô hình xác minh phần mềm***

## 1.2. ĐỊNH NGHĨA KIỂM CHỨNG MÔ HÌNH

*Định nghĩa 1:* Kiểm chứng mô hình là một kỹ thuật tự động, đưa ra một mô hình hữu hạn trạng thái của hệ thống và thuộc tính hình thức, hệ thống kiểm tra xem thuộc tính này có thỏa mãn cho mô hình.

*Định nghĩa 2:* Kiểm chứng mô hình là một kỹ thuật tự động để xác minh các thuộc tính, hành vi của một mô hình hệ thống, bằng cách duyệt tất cả các trạng thái của hệ thống đó.

*Định nghĩa 3:* Kiểm chứng mô hình là một kỹ thuật hiệu quả để phơi bày những lỗi tiềm ẩn ở giai đoạn thiết kế của hệ thống phần mềm [4].

## 1.3. KIỂM CHỨNG MÔ HÌNH PHẦN MỀM

Kiểm chứng mô hình phần mềm có hai ý nghĩa chính:

Đầu tiên, kiểm chứng mô hình phần mềm với mục đích chính là kiểm thử, xác minh xem hệ thống có thoả mãn một số thuộc tính, tính chất nào đó hay không. Khi đó, hệ thống được biểu diễn dưới dạng đồ thị các trạng thái, gọi là mô hình, các trạng thái này được liên kết với nhau bởi các bước chuyển trạng thái. Mỗi bước chuyển trạng thái tương ứng với một bước của chương trình được biểu diễn bằng toán học ngữ nghĩa hoặc ngôn ngữ máy. Các thuộc tính của phần mềm sẽ được kiểm chứng bằng cách duyệt toàn bộ đồ thị.

Tiếp theo, kiểm chứng mô hình phần mềm còn mang ý nghĩa logic tính toán nhằm kiểm tra xem hệ thống phần mềm có thể biểu diễn dưới dạng một mô hình công thức logic thời gian (temporal logic) hay không? Do đó, từ mô hình không chỉ mang ý nghĩa là việc đặc tả hành vi một cách trừu tượng mà còn là biểu diễn hành vi của hệ thống.

Trong kiểm chứng mô hình phần mềm, các thuộc tính cần thoả mãn được biểu diễn bằng logic thời gian hoặc bằng các Ôtômat. Sau đó, sẽ thực hiện phép duyệt toàn bộ không gian trạng thái để kiểm chứng xem hệ thống có thoả mãn các tính chất đó hay không. Vì vậy người ta gọi đó là kiểm chứng mô hình. Khi hệ thống và đặc tả của hệ thống được mô hình hoá bằng Ôtômat hữu hạn trạng thái, hệ thống sẽ được so sánh với đặc tả để kiểm tra xem các hành vi của hệ thống có phù hợp với đặc tả hay không.

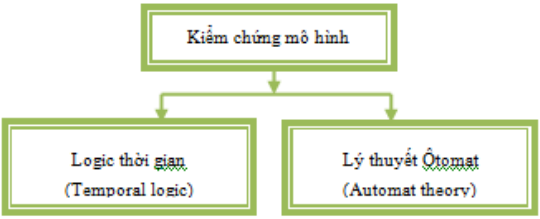
Do đó, kiểm chứng mô hình phần mềm còn được định nghĩa là một kỹ thuật tự động mà: khi cho một mô hình hữu hạn trạng thái của một hệ thống và một thuộc tính hệ thống cần thoả mãn, kiểm chứng xem hệ thống đó có thoả mãn thuộc tính đưa ra hay không?

## 1.4. CÁC PHƯƠNG PHÁP KIỂM CHỨNG MÔ HÌNH

### 1.4.1. Giới thiệu

Kiểm chứng mô hình dựa trên việc tạo ra mô hình hữu hạn của hệ thống và kiểm chứng mô hình đó với các thuộc tính đặt ra của phần mềm. Mô hình của hệ thống được biểu diễn dưới dạng máy

trạng thái hữu hạn. Sau đó, ta phải tìm cách để hoàn thành việc duyệt toàn bộ không gian trạng thái để kiểm chứng mô hình đó có thoả mãn với đặc tả hay không. Đặc tả hệ thống thường được biểu diễn dưới dạng logic thời gian hoặc Ôtômat, do đó sẽ có hai cách tiếp cận việc kiểm chứng mô hình: đó là kiểm chứng mô hình thời gian và kiểm chứng mô hình theo lý thuyết ôtômat (Hình 1-3).




***Hình 1-3: Các phương pháp tiếp cận kiểm chứng mô hình phần mềm***

Với bất cứ kỹ thuật kiểm chứng mô hình phần mềm nào đều phải giải quyết một vấn đề khó khăn nhất đó là: bùng nổ không gian trạng thái. Không gian trạng thái của việc kiểm chứng mô hình thường là tuyến tính nhưng không gian trạng thái của hệ thống lại thường tăng theo hàm mũ. Do đó, thách thức kỹ thuật chủ yếu trong việc kiểm chứng mô hình là thiết kế các phương thức và các cấu trúc dữ liệu để giải quyết được không gian trạng thái lớn như vậy. Có một số phương pháp để có thể tránh sự bùng nổ trạng thái, trong đó có 4 phương pháp chính (Hình 1-4).


***Hình 1-4: Phương pháp tiếp cận điều khiển sự bùng nổ không gian trạng thái***

#### 1.4.2. Phương pháp biểu diễn ký hiệu (Symbolic representation)

#### 1.4.3. Phương pháp duyệt nhanh

#### 1.4.4. Phương pháp rút gọn

***a. Rút gọn bậc từng phần***

***b. Tối thiểu hóa kết cấu***

***c. Trừu tượng hóa***

#### 1.4.5. Phương pháp xác minh kết cấu

#### 1.4.6. Nhận xét các phương pháp kiểm chứng mô hình

## 1.5. ƯU ĐIỂM VÀ NHƯỢC ĐIỂM CỦA KIỂM CHỨNG MÔ HÌNH

### 1.5.1. Ưu điểm của kiểm chứng mô hình

### 1.5.2. Nhược điểm của kiểm chứng mô hình

## 1.6. KIỂM CHỨNG MÔ HÌNH CỔ ĐIỂN VÀ HIỆN ĐẠI

## 1.7. SỰ KHÁC NHAU GIỮA KIỂM CHỨNG MÔ HÌNH PHẦN MỀM VÀ KIỂM THỬ PHẦN MỀM

Cả kiểm chứng mô hình và kiểm thử phần mềm đều thực hiện vai trò đảm bảo chất lượng phần mềm bằng việc tìm ra các lỗi nếu có của phần mềm. Nhưng giữa kiểm chứng mô hình và kiểm thử phần mềm có một số điểm khác nhau quan trọng sau:

- ✓ Kiểm thử phần mềm đòi hỏi phải có chương trình để thực hiện, còn kiểm chứng mô hình thì ngoài kiểm thử trên mã nguồn còn có thể dùng để kiểm chứng bản thiết kế, nghĩa là khi chương trình vẫn còn trên giấy.
- ✓ Kiểm thử phần mềm chỉ có thể khẳng định được chương trình không gặp lỗi đối với các trường hợp kiểm thử đã kiểm tra tức không tìm thấy lỗi chứ không khẳng định được là chương trình hoàn toàn không còn lỗi. Ngược lại, kiểm chứng phần mềm cho phép ta kết luận được chương trình hoàn toàn không còn lỗi.

- ✓ Trong thực tế kiểm thử phần mềm có ưu điểm rất lớn là dễ thực hiện. Một người bình thường cũng có thể thực hiện được. Trong khi đó, kiểm chứng mô hình đòi hỏi phải mô hình hóa và đặc tả, công việc này rất khó và đòi hỏi người thực hiện có trình, độ kinh nghiệm và kiến thức nhất định.

## 1.8. CÔNG CỤ KIỂM CHỨNG MÔ HÌNH

Để thực hiện kiểm chứng mô hình cho các hệ thống phần cứng hoặc phần mềm một cách tự động, người ta sử dụng phần mềm gọi là công cụ kiểm chứng mô hình. Phần mềm này giúp kiểm chứng mô hình hoá hệ thống ở bước thiết kế có đúng hoặc sai. Trong mục này, chúng tôi trình bày số công cụ kiểm chứng tiêu biểu.

### 1.8.1. Công cụ kiểm chứng mô hình NuSMV

### 1.8.2. Công cụ kiểm chứng mô hình SPIN

### 1.8.3. Công cụ kiểm chứng mô hình PathFinder

### 1.8.4. Công cụ kiểm chứng mô hình SLAM

## 1.9. KẾT CHƯƠNG

Trong chương 1, luận văn đã trình bày khá cơ bản về cơ sở lý thuyết liên quan đến đề tài, cụ thể gồm:

- ✓ Giới thiệu về lịch sử, khái niệm, ưu điểm và nhược điểm của kiểm chứng mô hình.
- ✓ Trình bày phương pháp, qui trình kiểm chứng mô hình phần mềm.
- ✓ Sự khác nhau giữa kiểm chứng mô hình và kiểm thử.
- ✓ Giới thiệu các công cụ kiểm chứng mô hình tiêu biểu.

Những nội dung trên cung cấp nền tảng lý thuyết quan trọng trong quá trình phát triển đề tài luận văn.

# CHƯƠNG 2 -
# HỆ THỐNG PHẢN ỨNG VÀ MÔI TRƯỜNG LUSTRE/SCADE

## 2.1. TỔNG QUAN HỆ THỐNG PHẢN ỨNG

### 2.1.1. Các định nghĩa

*Định nghĩa 1:* Hệ thống phản ứng là một hệ thống thay đổi hành động của mình với đầu ra, điều kiện và trạng thái nhằm đáp ứng với các tác động từ bên ngoài hệ thống. Hệ thống này có thể tự định hướng hoặc được điều khiển định hướng để phản ứng lại với các tác động bên ngoài.

Điều kiện đầu vào của một hệ thống phản ứng luôn không được sẵn sàng, tức là, luôn không thể biết trước chính xác đầu vào, khác với việc cộng trừ 2 số luôn biết được đầu vào. Một loại quan trọng nhất của hệ thống phản ứng là tương tác hệ thống. Các hệ thống này có thể phản ứng lại với các sự kiện bằng cách cung cấp đầu ra cho người sử dụng và lịch sử hoạt động của nó. Sản phẩm đầu ra có thể phản ứng với các sự kiện lịch sử hoặc dấu hiệu về tình trạng hệ thống[18].

*Định nghĩa 2:* Hệ thống phản ứng là hệ thống máy tính liên tục phản ứng lại với các tác động từ môi trường, khi môi trường này không thể đồng bộ hóa một cách hợp lý với hệ thống. Nói cách khác, môi trường không thể chờ đợi và hệ thống phải tôn trọng nghiêm ngặt ràng buộc về thời gian thực cần thiết đáp ứng kịp thời [11].

### 2.1.2. Đặc điểm của hệ thống phản ứng

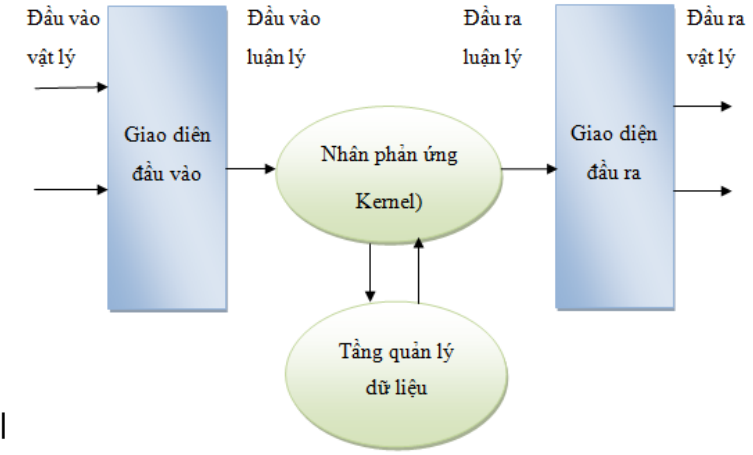
Các hệ thống phản ứng thông thường áp dụng chủ yếu cho các hệ thống điều khiển tự động, hệ thống liên quan đến giao diện người-máy với thời gian đáp ứng là ít. Nhìn chung, các hệ thống này có các đặc điểm sau:

- ✓ Song song (Parallelism): chúng hoạt động song song liên quan đến môi trường. Các hệ thống này có thể được chia ra thành quá trình đồng thời phụ và hợp tác với nhau.
- ✓ Các ràng buộc thời gian thời gian (Time constraints): đặc điểm này tuân thủ qui chế về thời gian nghiêm ngặt như tần số đầu vào và thời gian đáp ứng đầu vào-đầu ra.
- ✓ Thuyết định mệnh (Determinism): chúng tạo ra trình tự đầu ra giống với chuỗi trình tự đầu vào.

### 2.1.3. Kiến trúc của hệ thống phản ứng

Thông thường các chương trình hệ thống phản ứng bao gồm ba lớp (Hình 2-1):

- ✓ Giao diện vào/ra (Interface input/Output) : với môi trường đó là tiếp nhận đầu vào và đầu ra.
- ✓ Nhân phản ứng (Reactive kernel): chứa logic hoặc hành vi của hệ thống.
- ✓ Tầng xử lý dữ liệu (Data handling layer): thực hiện tính toán cổ điển theo yêu cầu của nhân phản ứng.




***Hình 2-1: Kiến trúc hệ thống phản ứng***

### 2.1.4. Các lĩnh vực ứng dụng

Hệ thống phản ứng có trong các lĩnh vực ứng dụng sau:

- ✓ Hệ thống điều khiển nhúng có trong: máy điều hòa, tủ lạnh, điện thoại, Ôtô, máy bay.
- ✓ Hệ thống điều khiển tiến trình: hệ điều khiển máy móc, rô-bốt…
- ✓ Hệ điều hành máy tính và mạng.
- ✓ Hệ thống giao diện người dùng.

## 2.2. NGÔN NGỮ ĐỒNG BỘ

Một ngôn ngữ được gọi là đồng bộ nếu đầu ra là đồng bộ hóa với các đầu vào. Ngôn ngữ như Lustre, Esterel và Argos được gọi là ngôn ngữ đồng bộ. Nói cách khác, chương trình sẽ phản ứng lại ngay lập tức trên một sự kiện, và đầu ra sẽ thay đổi ngay lập tức sau khi đầu vào thay đổi. Điểm đặc trưng cho một ngôn ngữ đồng bộ là sử dụng một khái niệm đa hình của thời gian, điều này có nghĩa là thời gian vật lý sẽ được xử lý như một sự kiện bên ngoài và tất cả các sự kiện có thể được sử dụng như là một kích hoạt đồng hồ. Những khía cạnh quan trọng là thời gian mà hai sự kiện có thể xảy ra đồng thời và thứ tự các sự kiện là như nhau ở tất cả các thời gian.

Các chương trình đồng bộ được sử dụng trong các hệ thống phản ứng. Hệ thống phản ứng với đầu vào từ môi trường nơi mà tốc độ đầu vào được xác định bởi môi trường. Một hệ thống phản ứng không bao giờ có thể phát sinh một sự kiện nếu môi trường không phản ứng. Hệ thống là tĩnh giữa các sự kiện, các mã thực thi được tạo ra bởi một ngôn ngữ đồng bộ luôn luôn là tuần tự, do vậy ngăn ngừa các vấn đề chia sẻ. Chỉ có một nhiệm vụ được thực hiện tại một thời điểm và nhiệm vụ đó sẽ được thực hiện cho đến khi nó được hoàn tất.

## 2.3. MÔ HÌNH LUỒNG DỮ LIỆU

Một cách tiếp cận để xây dựng các chương trình đồng bộ là mô hình luồng dữ liệu. Chương trình được xây dựng từ các toán tử kết hợp với nhau, làm vệc song song với nhau. Ngay sau khi một toán tử nhận được một đầu vào, nó tính toán đầu ra. Mô hình luồng

dữ liệu đã và đang được sử dụng chủ yếu trong lĩnh vực điều khiển và các lĩnh vực điện tử.

Mô hình luồng dữ liệu là một mô hình chức năng và rõ ràng về mặt toán học, làm cho phương pháp tiếp cận hiệu quả cho việc sử dụng các phương pháp chính thức để phân tích, thiết kế và xác minh.

## 2.4. NGÔN NGỮ LUSTRE

### 2.4.1. Giới thiệu

Lustre là một ngôn ngữ đồng bộ luồng dữ liệu hình thức, được thiết kế năm 1984 bởi Viện IMAG tại Grenoble. Được dùng để lập trình cho các đặc tả của những ứng dụng đồng bộ và thời gian thực như các ứng dụng của hệ thống phản ứng. Lustre đảm bảo sinh mã hiệu quả và cung cấp những đặc tả và các cơ sở xác minh hình thức.

### 2.4.2. Các kiểu dữ liệu cơ bản

### 2.4.3. Khai báo biến và chú thích

### 2.4.4. Nodes, Flows và Cycle

## 2.5. CẤU TRÚC CHƯƠNG TRÌNH

## 2.6. CÔNG CỤ LUKE

### 2.6.1. Giới thiệu

Luke là một công cụ để xử lý các tập tin Lustre. Nó chứa một mô phỏng, công cụ xác minh và đồng thời là trình biên dịch của cho Lustre [27].

### 2.6.2. Các ví dụ chương trình Lustre/Luke

## 2.7. MÔI TRƯỜNG SCADE

### 2.7.1. Giới thiêu

SCADE (Safety Critical Application Development Environment) là một công cụ đồ họa dành riêng để phát triển các hệ

thống nhúng quan trọng, hệ thống phản ứng và thường được sử dụng bởi các ngành công nghiệp. SCADE là một môi trường đồ họa dựa trên ngôn ngữ Lustre và nó cho phép định nghĩa phân cấp của các thành phần hệ thống và sinh mã tự động (có thể sinh mã chương trình c, Ada).

SCADE được sử dụng nhiều nhất trong các lĩnh vực hàng không vũ trụ và Quốc phòng, tuy nhiên với khả năng của SCADE cũng đáp ứng trong giao thông vận tải, Ô tô, năng lượng [26].

### 2.7.2. Các ứng dụng trong công nghiệp

### 2.7.3. Giới thiệu dòng sản phẩm SCADE Suite®

***a. Giới thiệu***

***b. Các tính năng của SCADE®***

## 2.8. KẾT CHƯƠNG

Trên cơ sở lý thuyết đã nghiên cứu ở chương trước, chương hai này tiếp tục trình bày các vấn đề cần thiết không thể thiếu để đi vào thực nghiệm trong chương 3 bao gồm:

- ✓ Tìm hiểu về cơ bản hệ thống phản ứng.
- ✓ Tìm hiểu về ngôn ngữ Lustre và cài đặt ví dụ cơ sở. Lustre là một ngôn ngữ lập trình cho các hệ thống phản ứng và ngôn ngữ chính để xây dựng ứng dụng kiểm chứng trong chương 3.
- ✓ Môi trường SCADE rất hữu dụng để phát triển các hệ thống phản ứng quan trọng và phức tạp. Việc triển khai ứng dụng trên môi trường SCADE nhằm đảm bảo chất lượng phần mềm ở mức cao nhất có thể, giảm thiểu thời gian phát triển, đảm bảo sự tin cậy gần như tuyệt đối.

# CHƯƠNG 3 -
# ỨNG DỤNG KIỂM CHỨNG MÔ HÌNH CHO CÁC HỆ THỐNG PHÁT TRIỂN TRÊN MÔI TRƯỜNG LUSTRE/SCADE

## 3.1. PHÂN TÍCH VÀ LỰA CHỌN CÔNG CỤ KIỂM CHỨNG MÔ HÌNH

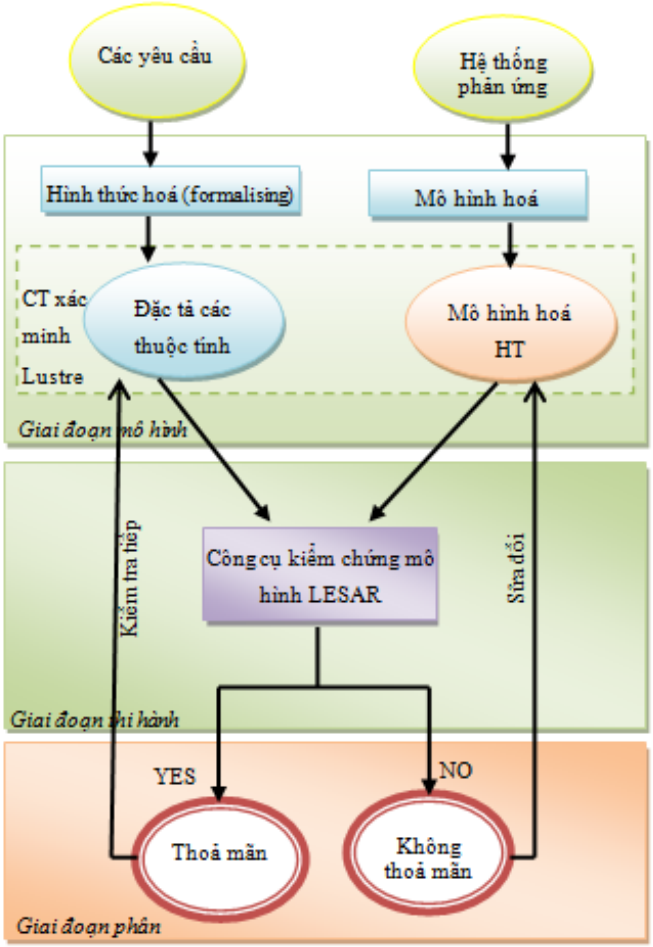
## 3.2. QUY TRÌNH KIỂM CHỨNG MÔ HÌNH CHO HỆ THỐNG PHẢN ỨNG VỚI LESAR

Tiến trình kiểm chứng mô hình áp dụng ở mức thiết kế cho các hệ thống phản ứng được phân chia thành 3 giai đoạn phân biệt sau:

- ✓ Giai đoạn mô hình hoá (Modeling phase): đầu tiên mô hình hóa cho hệ thống phản ứng sử dụng ngôn ngữ mô tả Lustre thực thi trên công cụ kiểm chứng Lesar. Đặc tả các thuộc tính hình thức của hệ thống phản ứng với ngôn ngữ Lustre. Như vậy giai đoạn mô hình hoá này, chung tôi đều sử dụng ngôn ngữ Lustre và được viết trong một tệp tin chương trình. Nhận thấy rằng ở gian đoạn này chính là xây dựng đầu vào của quá trình kiểm chứng. Kế tiếp đặc tả giả định môi trường của hệ thống phản ứng với ngôn ngữ Lustre.
- ✓ Giai đoạn thi hành (Running phase): giai đoạn này thực thi công cụ kiểm chứng mô hình Lesar để kiểm tra tính hợp lệ của các tính chất trong mô hình hoá hệ thống có trong chương trình Lustre. Khi thực thi với Lesar thường thiết lập các tuỳ chọn như thuật toán kiểm chứng, tuỳ chọn để xem chi tiết số các nút, giao tác được tạo ra trong quá trình xác minh Lesar. Có hai kỹ thuật xác minh đã được cài đặt và tích hợp trong công cụ kiểm chứng Lesar đó là kỹ thuật liệt kê tường minh (explicitly enumerates) các trạng thái có thể truy cập và được sử dụng tiêu biểu trong công cụ kiểm chứng mô

hình. Giới hạn chính của cách tiếp cận này là rõ ràng số trạng thái cần được xem xét. Kỹ thuật thứ hai là tính toán tượng trưng (symbolic computations) dựa trên công thức sử dụng lược đồ quyết định nhị phân BDD. Hướng tiếp cần này còn được gọi là "Symbolic model checking".

- ✓ Giai đoạn phân tích (Analysis phase): thực nghiệm cho thấy, có thể có ba loại kết quả đầu ra sau khi thực thi chương trình kiểm chứng với Lesar: khi thực thi chương trình xác minh trên công cụ kiểm chứng mô hình Lesar. Nếu thuộc tính đầu tiên được kiểm tra thoả mãn khi duyệt qua tất cả không gian trạng thái của hệ thống, và tiếp tục kiểm tra tuần tự các thuộc tính đã được đặc tả trong trương trình Lustre. Khi công cụ Lesar thực hiện xác minh nếu thoả mãn tất cả các thuộc tính được đặc tả kết quả trả về đúng (Yes). Như vậy hệ thống đã kiểm chứng là đúng ở mức thiết kế. Trường hợp này được xem là hệ thống không lỗi so với đặc tả và giả định được đặt ra. Nếu một thuộc tính được Lesar phát hiện sai (No), có nghĩa là việc mô hình hoá ở bước này chưa đúng với thuộc tính được đặt ra. Chúng ta cần phân tích nguyên nhân phát sinh vấn đề không hợp lệ này, tiếp theo quay lại giai đoạn mô hình hoá để thực hiện sửa đổi mô hình hệ thống. Sau đó thực hiện kiểm chứng lặp lại với công cụ Lesar. Một trường hợp khác xảy ra là tràng bộ nhớ. Như vậy chúng ta cần phải rút gọn mô hình (cải tiến việc mô hình hoá hệ thống phản ứng) và thử chạy lại Lesar.



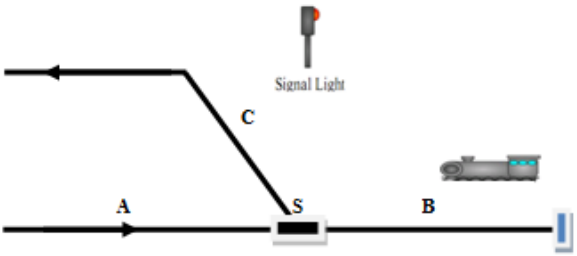

***Hình 3-1: Qui trình kiểm chứng mô hình cho hệ thống phản ứng với công cụ Lesar***

## 3.3. THỬ NGHIỆM KIỂM CHỨNG MÔ HÌNH

### 3.3.1. Kiểm chứng mô hình cho hệ thống tàu điện ngầm UMS

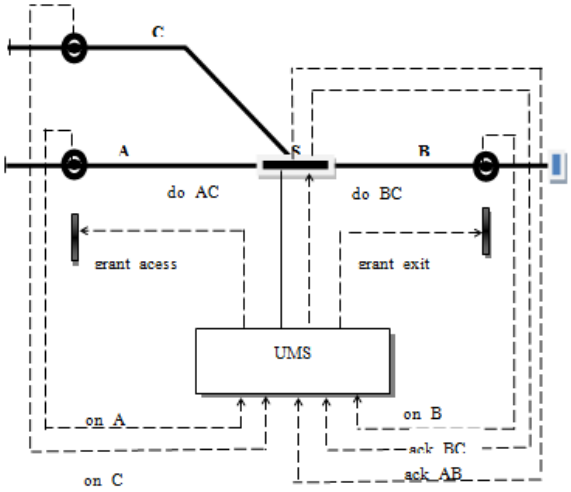
***a. Giới thiệu hệ thống tàu điện ngầm UMS***

Chúng tôi sẽ giới thiệu một ví dụ được chuyển thể từ một thiết bị tàu điện ngầm. Tại kết thúc của một đường tàu điện ngầm, đoạn "U-turn" đặc biệt cho phép các tàu chuyển từ đoạn rây này sang đoạn rây khác. Và quay trở lại theo hướng ngược lại (Hình 3-2). Đoạn "U-turn" là sự kết hợp bởi 3 đường rây A,B,C và bộ chuyển S. Giả sử rằng đường rây vào A và đường rây thoát ra là C. Tàu chuyển từ A tới C đầu tiên phải chờ S kết nối. A với B, sau đó B quá cảnh và chờ đợi một lần nữa cho S để kết nối B với C trước khi quay trở lại trên đường rây C.




***Hình 3-2: Đoạn "U-turn" của tàu điện ngầm***

Tổng quan về hệ thống và môi trường của hệ thống này được thể hiện (Hình 3-3).




***Hình 3-3: Hệ thống tàu điện ngầm UMS và môi trường của hệ thống***

***b. Biểu diễn các thuộc tính then chốt (critical properties)***

***c. Các thuộc tính an toàn của hệ thống***

***d. Mô hình hóa các giả định hành vi môi trường***

***e. Chương trình xác minh với Lesar***

***f. Thực thi chương trình với Lesar***

***g. Nhận xét kết quả thử nghiệm***

Sau khi thực thi chương trình xác minh cho hệ thông UMS trên Lesar với 4 thuộc tính an toàn đã được cài đặt trong chương trình Lustre. Kết quả nhận xét như sau:

- ✓ Từng thuộc tính đã được thỏa mãn đúng với mô hình hóa hệ thống tàu điện ngầm UMS được mô tả. Như vậy quá trình thử nghiệm kiểm chứng mô hình cho hệ thống tàu điện ngầm UMS đã được thỏa mãn.

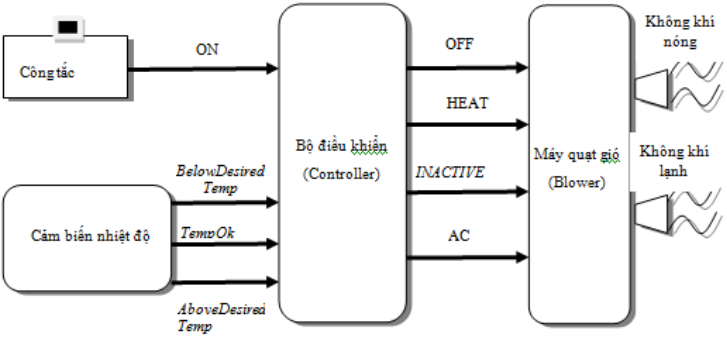
- ✓ Bên cạnh đó chúng tôi nhận thấy rằng, tùy thuộc vào giải thuật chọn lựa để xác minh trên Lesar mà cho ra các thông số về: số trạng thái thực hiên, số bước chuyển trạng thái và tổng số bộ nhớ cần thiết cho BDD lại khác nhau (Bảng 3-3). Tất nhiên, kết quả đầu ra của quá trình xác minh là không đổi.
- ✓ Với giải thuật Enumerative trên Lesar cho thấy số trạng thái là 49 và bước chuyển trạng thái là 263, như vậy là nhiều và phức tạp hơn so với giải thuật Symbolic forward có số trạng thái là 9. Nhưng với kỹ thuật Enumerative thì tổng số bộ nhớ BDD 67.54 KB nhỏ hơn một nữa so với kỹ thuật Symbolic forward xấp xĩ 147.98 KB.

**Bảng 3-1: So sánh kết quả xác minh hệ thống với từng giải thuật**

| Giải thuật xác minh Lesar | Trạng thái thực hiện | Bước chuyển trạng thái | Tổng số bộ nhớ BDD | Kết quả đầu ra |
|---|---|---|---|---|
| **Enumerative** | 49 | 263 | 665 nút (67.54KB) | True Property |
| **Symbolic forward** | 9 | | 1457 nút (~147.98 KB) | True Property |
| **Symbolic backward** | | | | Non causal Assertions |

### *3.3.2.* Kiểm chứng mô hình cho hệ thống điều khiển nhiệt độ

#### *a. Giới thiệu hệ thống*




***Hình 3-4: Kiến trúc của hệ thống điều khiển nhiệt độ***

#### *b. Mô hình hóa các giả định hành vi môi trường*

#### *c. Các thuộc tính an toàn của hệ thống*

#### *d. Chương trình xác minh với Lesar*

## 3.4. KẾT CHƯƠNG

Dựa trên nền tảng về cơ sở lý thuyết kiểm chứng mô hình ở chương 1 và tổng quan về hệ thống phản ứng, ngôn ngữ lập trình luồng dữ liệu Lustre, môi trường SCADE, bộ phần mềm SCADE Suite® cùng với các ưu điểm, các lĩnh vực ứng dụng ở chương 2. Chương 3 của luận văn đã phân tích và lựa chọn công cụ kiểm chứng mô hình phù hợp cho các hệ thống phản ứng. Kế tiếp, đã nêu lên qui trình các bước cụ thể kiểm chứng mô hình tương ứng cho hệ thống phản ứng nằm trong phạm vi kiểm chứng của đề tài. Đặc biệt đã thử nghiệm thành công trên hai ứng dụng về hệ thống phản ứng: hệ thống quản lý tự động đoạn "U-turn" tàu điện ngầm và hệ thống điều khiển nhiệt độ.

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## 1. Kết quả đạt được của luận văn

Đề tài ***"Nghiên cứu và ứng dụng kiểm chứng mô hình cho các hệ thống phát triển trên môi trường Lustre/SCADE"*** của chúng tôi đã hoàn thành và đáp ứng được các yêu cầu đặt ra với những kết quả như sau:

- ✓ Trình bày các vấn đề cơ bản về kiểm chứng mô hình bao gồm: khái niệm, phương pháp và qui trình các bước kiểm chứng mô hình ở mức khái quát.
- ✓ Trình bày tổng quan về ngôn ngữ lập trình đồng bộ Lustre/SCADE, bao gồm: các khái niệm liên quan, cấu trình viết chương trình và các ví dụ minh họa cụ thể về một số chương trình Lustre cơ bản.
- ✓ Giới thiệu tổng quan môi trường SCADE, vai trò và ứng dụng môi trường SCADE phát triển các hệ thống thiên về phần mềm nhúng, hệ thống phản ứng.
- ✓ Nghiên cứu và giới thiệu bộ phần mềm SCADE Suite®, phần mềm này chính là công nghệ để xây dựng các hệ thống phần mềm hiện đại với yêu cầu đảm bảo sự tin cậy, an toàn gần ở mức cao nhất (hệ thống quản lý lò phản ứng hạt nhân, hệ thống điều khiển tầu điện ngầm, … vv).
- ✓ Phân tích và lựa chọn công cụ kiểm chứng mô hình Lesar dùng để xác minh cho hệ thống phản ứng.
- ✓ Xây dựng được qui trình kiểm chứng mô hình cho hệ thống phản ứng, sử dụng ngôn ngữ Lustre và xác minh bởi công cụ kiểm chứng mô hình Lesar.
- ✓ Thử nghiệm kiểm chứng mô hình sử dụng công cụ kiểm chứng Lesar để xác minh cho 2 hệ thống cụ thể: hệ thống quản lý tự động đoạn "U-turn" tàu điện ngầm và hệ thống điều khiển nhiệt độ.

## 2. Hạn chế của luận văn

Thực tế, chúng tôi đã có nhiều cố gắng và nỗ lực nghiên cứu trong thời gian cho phép, tuy nhiên trong luận văn vẫn không tránh khỏi những thiếu sót:

- ✓ Đề tài có nội dung đề cập hướng nghiên cứu mới, lĩnh vực nghiên cứu chưa phổ biến đối với sinh viên cao học nên việc tham khảo, áp dụng có gặp nhiều khó khăn.
- ✓ Lý thuyết kiểm chứng mô hình rất phức tạp, tài liệu phần lớn bằng tiếng anh và thời gian nghiên cứu chưa nhiều. Vì vậy sự hiểu biết còn hạn chế và cần phải học nhiều hơn nữa.
- ✓ Đề tài đã được thử nghiệm kiểm chứng mô hình áp dụng cho những hệ thống phản ứng. Vì vậy chưa được thử nghiệm cho các phần mềm phổ biến được viết hiện nay.

## 3. Hướng phát triển

Hướng phát triển tiếp của đề tài này có thể:

- ✓ Mở rộng nghiên cứu chuyên sâu về kiểm chứng mô hình trên các hệ thống phần cứng, phần mềm hiện đại.
- ✓ Mở rộng kiểm chứng mô hình, thử nghiệm cho các môi trường phát triển khác như hệ thống rút tiền ATM, hệ thống báo động trong các ngôi nhà thông minh.
- ✓ Xây dựng công cụ kiểm chứng mô hình phần mềm sử dụng ngôn ngữ lập trình thông dụng để mô hình hoá như C, Java, C#.
- ✓ Một hướng khác là xây dựng công cụ mô hình hóa tự động bản thiết kế phần mềm.