TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

──────── \* ───────

ĐỒ ÁN

**TỐT NGHIỆP ĐẠI HỌC**

NGÀNH CÔNG NGHỆ THÔNG TIN

**ỨNG DỤNG**

**MÔ HÌNH PHAM-NORDMANN-ZHANG  
VÀO DỰ ĐOÁN TÍNH TIN CẬY**

**CHO CÁC HỆ THỐNG PHẦN MỀM**

**HƯỚNG THÀNH PHẦN**

Sinh viên thực hiện : **Phạm Bình**

Lớp IS2 – Việt Nhật – K55

Giáo viên hướng dẫn : PGS.TS **Huỳnh Quyết Thắng**

HÀ NỘI 5-2015

**PHIẾU GIAO NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP**

1. Thông tin về sinh viên

Họ và tên sinh viên: . Phạm Bình

Điện thoại liên lạc: 01663725615 Email: binh.pham92@gmail.com

Lớp: IS2 – Việt Nhật – K55 Hệ đào tạo: Đại học chính quy

Đồ án tốt nghiệp được thực hiện tại:

Thời gian làm ĐATN: Từ ngày 12/1/2015 đến 30/5/2015

2. Mục đích nội dung của ĐATN

* Nghiên cứu cách dự đoán tính tin cậy của các hệ thống phần mềm hướng thành phần.
* Nghiên cứu lý thuyết mô hình Pham-Nordmann-Zhang.
* Phát triển thành công công cụ dự đoán tính toán tin cậy cho các hệ thống hướng thành phần.

3. Các nhiệm vụ cụ thể của ĐATN

* Tìm hiểu cách dự đoán tính tin cậy dựa trên cơ sở hướng thành phần.
* Tìm hiểu mô hình Pham-Nordmann-Zhang.
* Tìm hiểu cách áp dụng mô hình Pham-Nordmann-Zhang vào dự đoán tính tin cậy cho các hệ thống phần mềm hướng thành phần.
* Tìm hiểu về cách nâng cao tính tin cậy bằng sử dụng công nghệ cấu trúc chịu lỗi.
* Phát triển công cụ thực tế (Công cụ RMPI - Reliability Modeling, Prediction, and Improvements).

4. Lời cam đoan của sinh viên:

Tôi – *Phạm Bình* - cam kết ĐATN là công trình nghiên cứu của bản thân tôi dưới sự hướng dẫn của *PGS. TS. Huỳnh Quyết Thắng* *và Ths. Nguyễn Hùng Cường.*

Các kết quả nêu trong ĐATN là trung thực, không phải là sao chép toàn văn của bất kỳ công trình nào khác.

|  |  |
| --- | --- |
|  | *Hà Nội, ngày 28 tháng 5 năm 2015*  Tác giả ĐATN  *Phạm Bình* |

5. Xác nhận của giáo viên hướng dẫn về mức độ hoàn thành của ĐATN và cho phép bảo vệ:

|  |  |
| --- | --- |
|  | *Hà Nội, ngày 28 tháng 5 năm 2015*  Giáo viên hướng dẫn  *PGS. TS. Huỳnh Quyết Thắng* |

# Lời nói đầu

Đầu tiên, cho phép tôi được gửi lời cảm ơn đến các thầy, các cô thuộc Viện Công Nghệ Thông Tin và Truyền Thông, trường Đại Học Bách Khoa Hà Nội đã truyền thụ những kiến thức, kinh nghiệm quý báu, đồng thời tạo điều kiện giúp tôi hoàn thành công việc nghiên cứu thực hiện đồ án tốt nghiệp.

Đặc biệt, tôi xin chân thành cảm ơn các thầy giáo – Phó giáo sư, tiến sĩ Huỳnh Quyết Thắng và NCS. Ths. Nguyễn Hùng Cường – những người đã trực tiếp hướng dẫn, chỉ bảo tận tình, giúp đỡ tôi suốt quá trình thực hiện, hoàn thành tốt đồ án.

Qua đây, tôi cũng xin cảm ơn gia đình và bạn bè đã động viên, đóng góp ý kiến giúp tôi hoàn thành đồ án.

|  |  |
| --- | --- |
|  | *Hà Nội, ngày 28 tháng 5 năm 2015*  Tác giả ĐATN  *Phạm Bình* |

# TÓM TẮT NỘI DUNG ĐỒ ÁN TỐT NGHIỆP

Khoa học máy tính đã và đang phát triển rất nhanh trong những thập kỷ gần đây. Các hệ thống máy tính đang được ứng dụng vào rất nhiều lĩnh vực, với mức độ phức tạp và yêu cầu về sự an toàn càng ngày càng cao. Các ứng dụng trên các hệ thống đang phát triển rộng rãi: hệ thống kiểm soát không lưu, nhà máy năng lượng hạt nhân, máy bay, hệ thống quân sự sử dụng thời gian thực, hệ thống thanh toán tự động trong ngân hàng, hệ thông giám sát bệnh nhân trong bệnh viện…

Kích thước và độ phức tạp của các hệ thống máy tĩnh cũng ngày càng tăng: từ bộ đơn xử lý đến bộ đa xử lý phân tán, từ hệ thống cá nhân tách biệt đến hệ thống mạng tích hợp, từ chương trình quy mô nhỏ đến chương trình chia sẻ tài nguyên quy mô lớn…Điều này đồng nghĩa với việc thiết kế hệ thống, hay phân bổ nguồn lực cũng ngày càng trở nên khó khăn.

Không có phương pháp phổ biến nào để đánh giá các hệ thống máy tính. Khi đó, độ tin cậy là một thước đo định lượng hữu ích giúp người sử dụng đánh giá một hệ thống máy tính có tốt hay không.

Trong thực tế, để tăng hiệu suất của hệ thống máy tính và cải thiện quá trình phát triển, phân tích kỹ lưỡng về độ tin cậy là một công việc cần thiết. Dựa trên các mô hình và sự phân tích, các phương pháp để cải thiện độ tin cậy hệ thống có thể được đưa vào sử dụng.

Đó là nguyên nhân, trong đồ án này, tối nghiên cứu về đề tài “Ứng dụng mô hình Pham-Nordmann-Zhang vào dự đoán tính tin cậy cho các hệ thống phần mềm hướng thành phần”. Tôi tập trung vào việc phát triển cách dự đoán tính tin cậy cho các hệ thống phần mềm hướng thành phần có ứng dụng mô hình Pham-Nordmann-Zhang và sau đó, cải thiện nâng cao tính tin cậy đó bằng công nghệ cấu trúc chịu lỗi. Hướng tiếp cận tôi tìm hiểu sử dụng mô hình hóa tính tính cậy, trong đó, các mô hình này được tự động chuyển đổi thành mô hình Pham-Nordmann-Zhang thông qua công cụ mà tôi phát triển.

# ABSTRACT OF THESIS

Computing has been the fastest developing technology during the last century. Computing systems are widely used in many areas, and they are desired to achieve various complex and safety-critical missions. The applications of the computing systems have now crossed many different fields and can be found in different products, for example, air traffic control systems, nuclear power plants, aircrafts, real-time military systems, telephone switching, bank auto-payment, hospital patient monitoring systems, and so forth.

The size and complexity of the computing systems has increased from one single processor to multiple distributed processors, from individual-separated systems to networked-integrated systems, from small-scale program running to large-scale resource sharing, and from local-area computation to global-area collaboration. As a result, making decisions in the system design or resource allocation also becomes difficult accordingly.

There is no common approach to assess computing systems. Reliability is a quantitative measure useful in this context as reliability can be broadly interpreted as the ability for a system to perform its intended function.

In fact, in order to increase the performance of the computing systems and to improve the development process, a thorough analysis of their reliability is needed. Based on the models and analysis, approaches to improve system reliability can be further implemented.

Because of that, I have been researching the topic of “Applying Pham-Nordmann-Zhang Model in Reliability Prediction for Component-based Systems”. I’m focus on evaluate reliability prediction of component-based systems by applying PNZ and then improving its reliability by using fault tolerance structures' technique. Our approach uses a reliability modeling schema whose models are automatically transformed by reliability prediction tool into PNZ models for reliability predictions and sensitivity analyses.

**Mục Lục**

[Lời nói đầu 4](#_Toc419988134)

[TÓM TẮT NỘI DUNG ĐỒ ÁN TỐT NGHIỆP 5](#_Toc419988135)

[ABSTRACT OF THESIS 6](#_Toc419988136)

[DANH MỤC HÌNH VẼ 9](#_Toc419988137)

[MỞ ĐẦU 12](#_Toc419988138)

[CHƯƠNG I: TỔNG QUAN 13](#_Toc419988139)

[1. Đặt vấn đề 13](#_Toc419988140)

[1.1. Bỏ qua loại lỗi lan tuyền 13](#_Toc419988141)

[1.2. Bỏ qua cơ chế chịu lỗi phần mềm 13](#_Toc419988142)

[1.3. Bỏ qua các lỗi đồng thời 13](#_Toc419988143)

[2. Mục tiêu và nhiệm vụ đồ án 14](#_Toc419988144)

[3. Định hướng giải quyết vấn đề 14](#_Toc419988145)

[3.1. Xem xét loại lỗi lan truyền 14](#_Toc419988146)

[3.2. Xem xét về cơ chế chịu lỗi phần mềm (FTMs) 15](#_Toc419988147)

[3.3. Xem xét lỗi đồng thời 15](#_Toc419988148)

[4. Kết chương 15](#_Toc419988149)

[CHƯƠNG II: DỰ ĐOÁN ĐỘ TIN CẬY VÀ MÔ HÌNH PNZ 17](#_Toc419988150)

[1. Dự đoán tính tin cậy dựa trên cơ sở hướng thành phần 17](#_Toc419988151)

[2. Mô hình Pham-Nordmann-Zhang (PNZ) 18](#_Toc419988152)

[3. Mô hình hóa tính tin cậy 20](#_Toc419988153)

[3.1. Bộ phận đặc tả tính tin cậy 20](#_Toc419988154)

[3.1.1. Các dịch vụ, các thành phần, các thực thi dịch vụ 20](#_Toc419988155)

[3.1.2. Các mô hình thất bại 22](#_Toc419988156)

[3.1.3. Các cấu trúc chịu lỗi 23](#_Toc419988157)

[3.2. Mô hình hóa tính tin cậy hệ thống 23](#_Toc419988158)

[4. Dự đoán tính tin cậy ứng dụng mô hình PNZ 24](#_Toc419988159)

[4.1. Chuyển đổi mỗi thành phần “UsageProfile” 24](#_Toc419988160)

[4.1.1. Cấu trúc tuyến tính 24](#_Toc419988161)

[4.1.2. Cấu trúc rẽ nhánh 25](#_Toc419988162)

[4.1.3. Cấu trúc vòng lặp 25](#_Toc419988163)

[4.1.4. Cấu trúc song song 26](#_Toc419988164)

[4.1.5. Cấu trúc thử lại 27](#_Toc419988165)

[4.1.6. Cấu trúc đa loại trừ 27](#_Toc419988166)

[4.2. Tổng kết các kết quả để đưa ra tính tin cậy hệ thống 28](#_Toc419988167)

[4.3. Cách thực thi 28](#_Toc419988168)

[5. Nâng cao tính tin cậy hệ thống phần mềm bằng cấu trúc chịu lỗi 29](#_Toc419988169)

[6. Kết chương 31](#_Toc419988170)

[CHƯƠNG III. PHÁT TRIỂN CÔNG CỤ MÔ HÌNH HÓA ĐỘ TIN CẬY RMPI 32](#_Toc419988171)

[1. Giới thiệu công cụ RMPI 32](#_Toc419988172)

[1.1. Giới thiệu chung 32](#_Toc419988173)

[1.2. Cấu trúc dữ liệu XML đầu vào 32](#_Toc419988174)

[1.3. Hình ảnh minh họa 34](#_Toc419988175)

[2. Phát triển công cụ RMPI 36](#_Toc419988176)

[2.1. Ứng dụng mô hình PNZ 36](#_Toc419988177)

[2.2. Phát triển giao diện đồ họa người dùng 40](#_Toc419988178)

[2.3. Hình ảnh minh họa 41](#_Toc419988179)

[3. Thử nghiệm và đánh giá công cụ 42](#_Toc419988180)

[3.1. Mô tả bài toán thử nghiệm 42](#_Toc419988181)

[3.1.1. Các dịch vụ, các thành phần, các thực thi của hệ thống 42](#_Toc419988182)

[3.1.2. Các mô hình thất bại của hệ thống 44](#_Toc419988183)

[3.1.3. Kiến trúc tổng quát của hệ thống 45](#_Toc419988184)

[3.2. Tính toán sử dụng công cụ RMPI 46](#_Toc419988185)

[3.3. Nhận xét, đánh giá công cụ RMPI 51](#_Toc419988186)

[4. Kết chương 52](#_Toc419988187)

[KẾT LUẬN 53](#_Toc419988188)

[1. Kết quả đạt được 53](#_Toc419988189)

[2. Hướng phát triển 54](#_Toc419988190)

[TÀI LIỆU THAM KHẢO 55](#_Toc419988191)

# DANH MỤC HÌNH VẼ

[Hình 1. Hướng dự đoán tính tin cậy dựa trên cơ sở hướng thành phần 17](#_Toc420530012)

[Hình 2. Sơ đồ các bộ phận đặc tả tính tin cậy 20](#_Toc420530013)

[Hình 3. Cấu trúc tuyến tính 21](#_Toc420530014)

[Hình 4. Cấu trúc rẽ nhánh 21](#_Toc420530015)

[Hình 5. Cấu trúc vòng lặp 21](#_Toc420530016)

[Hình 6. Cấu trúc song song 21](#_Toc420530017)

[Hình 7. Thành phần cấu trúc chịu lỗi sử dụng cấu trúc thử lại 30](#_Toc420530018)

[Hình 8. Thành phần cấu trúc chịu lỗi sử dụng cấu trúc đa loại trừ 30](#_Toc420530019)

[Hình 9. Màn hình chính giao diện dòng lệnh RMPI 34](#_Toc420530020)

[Hình 10. Màn hình kiểm tra đầu vào giao diện dòng lệnh RMPI (chính xác) 35](#_Toc420530021)

[Hình 11. Màn hình kiểm tra đầu vào giao diện dòng lệnh RMPI (lỗi) 35](#_Toc420530022)

[Hình 12. Màn hình dự đoán tính tin cậy giao diện dòng lệnh RMPI 36](#_Toc420530023)

[Hình 13. Khối PNZ 37](#_Toc420530024)

[Hình 14. Sơ đồ UML cho thành phần bổ sung của RMPI 40](#_Toc420530025)

[Hình 15. Sơ đồ cấu trúc giao diện đồ họa RMPI 40](#_Toc420530026)

[Hình 16. Màn hình chính giao diện đồ họa RMPI 41](#_Toc420530027)

[Hình 17. Màn hình sau khi mở file xml giao diện đồ họa RMPI 42](#_Toc420530028)

[Hình 18. Các thành phần của hệ thống WebScan 43](#_Toc420530029)

[Hình 19. Các thực thi dịch vụ của thành phần DocumentManager 43](#_Toc420530030)

[Hình 20. Các thực thi dịch vụ của thành phần WebScanControl 44](#_Toc420530031)

[Hình 21. Các thực thi dịch vụ của thành phần ClientReaction 44](#_Toc420530032)

[Hình 22. Cấu trúc hệ thống WebScan 45](#_Toc420530033)

[Hình 23. Màn hình kiểm tra đầu vào giao diện đồ họa RMPI 46](#_Toc420530034)

[Hình 24. Màn hình dự đoán tính tin cậy giao diện đồ họa RMPI 47](#_Toc420530035)

[Hình 25. Màn hình mô phỏng tính tin cậy (simulation) giao diện đồ họa RMPI 48](#_Toc420530036)

[Hình 26. Đồ thị tỷ lệ xuất hiện các loại lỗi(%) của hệ thống WebScan 48](#_Toc420530037)

[Hình 27. Mô hình module chịu lỗi 49](#_Toc420530038)

[Hình 28. Màn hình dự đoán tính tin cậy sau khi sử dụng cấu trúc chịu lỗi 50](#_Toc420530039)

DANH MỤC BẢNG

[Bảng 1. Các đặc tả sử dụng trong quá trình Poisson không đồng nhất 19](#_Toc420529993)

[Bảng 2. Các loại lỗi lan truyền của hệ thống WebScan 44](#_Toc420529994)

[Bảng 3. Các loại lỗi dừng của hệ thống WebScan 45](#_Toc420529995)

[Bảng 4. Các Instances cho các thành phần của hệ thống WebScan 45](#_Toc420529996)

[Bảng 5. Sai số giữa dự đoán tính tin cậy và mô phỏng tính tin cậy 48](#_Toc420529997)

[Bảng 6. Instance cho thành phần chịu lỗi của hệ thống WebScan 49](#_Toc420529998)

[Bảng 7. Bảng so sánh kết trả trước và sau khi sử dụng cấu trúc chịu lỗi FTSs 51](#_Toc420529999)

DANH MỤC THUẬT NGỮ

|  |  |  |
| --- | --- | --- |
| **Thuật ngữ** | **Tên đầy đủ** | **Ý nghĩa** |
| RMPI | Realiability, Modeling, Prediction and Improvements | Công cụ do TS. Phạm Thành Trung phát triển nền tảng ban đầu |
| PNZ | Pham-Nordmann-Zhang | Mô hình dự đoán tính tin cậy do TS. Phạm Hoàng và đồng nghiệp đề xuất |
| FTMs | Software Fault Tolerance Mechanisms | Cơ chế chịu lỗi phần mềm |
| FTSs | Software Fault Tolerance Structures | Cấu trúc chịu lỗi phần mềm |
| XML | EXtensible Markup Language | Ngôn ngữ đánh dấu mở rộng |
| XSD | XML Schema Definition | Tài liệu mô tả cấu trúc XML |
| DTMC | Discrete time Markov chain | Xích Markov trên trạng thái không gian rời rạc hoặc đếm được |
| CBSE | Component-based software engineering | Kỹ thuật phầm mềm dựa trên cơ sở hướng thành phần |
| IA | Internal Activity | Hoạt động nội bộ |
| PB | PNZ block | Khối PNZ |
| RP | Retry Part | Thành phần thử lại |
| MP | Multitrycatch Part | Thành phần đa loại trừ |
| Usage Profile | Usage Profile | Hồ sơ sử dụng |

# MỞ ĐẦU

Tính tin cậy là một trong tám đặc tính về chất lượng chính của các hệ thống phần mềm [1]. Đặc tính này được áp dụng cho rất nhiều giai đoạn của sản xuất phần mềm: phân tích, thiết kế, lập trình, kiểm thử. Có hai hướng để tiếp cận đặc tính này đó là: đánh giá [2][3][4] và dự đoán [5][6][7]. Trong đồ án này, tôi tập trung vào hướng thứ hai đó là dự đoán tính tin cậy hệ thống phần mềm. Tuy nhiên, Các hướng tiếp cận dự đoán tính tin cậy cho các hệ thống phần mềm hướng thành phần đã tồn tại đang bị giới hạn trong việc áp dụng vào thực tế, bởi họ đang bỏ qua hoặc ko hỗ trợ một cách rõ ràng cho các yếu tố như: lỗi lan truyền, cơ chế chịu lỗi phần mềm…

Trong đồ án này, tôi tập trung vào việc phát triển cách dự đoán tính tin cậy cho các hệ thống phần mềm hướng thành phần có ứng dụng mô hình Pham-Nordmann-Zhang và sau đó, cải thiện nâng cao tính tin cậy đó bằng công nghệ cấu trúc chịu lỗi. Hướng tiếp cận tôi tìm hiểu sử dụng mô hình hóa tính tính cậy, trong đó, các mô hình này được tự động chuyển đổi thành mô hình Pham-Nordmann-Zhang thông qua công cụ mà tôi phát triển. Đồng thời, tôi cũng có những mô phỏng thực tế, để cho thấy cách công cụ này dự đoán tính cũng cậy, cũng như nâng cáo tính tin cậy đó bằng việc sử dụng cơ chế chịu lỗi phần mềm.

Đồ án của tôi được chia là 3 chương chính:

CHƯƠNG I: TỔNG QUAN. Giới thiệu khái quát về các vấn đề mà hướng tiếp cận để dự đoán tính tin cậy hiện nay vẫn còn thiết sót. Từ đó đưa ra được mục tiêu và nhiệm vụ cụ thể cũng như phương hướng giải cho đồ án này.

CHƯƠNG II: DỰ ĐOÁN ĐỘ TIN CẬY VÀ MÔ HÌNH PNZ. Giới thiệu được toàn bộ các lý thuyết liên quan đến cách dự đoán tính tin cậy dựa trên cơ sở hướng thành phần, mô hình Pham-Nordmann-Zhang, cơ chế chịu lỗi phần mềm.

CHƯƠNG III: PHÁT TRIỂN CÔNG CỤ RMPI. Giới thiệu về công cụ tiền thân của RMPI, những thành phần mà tôi đã phát triển trong đồ án, cũng như thử nghiệm thực tế cách sử dụng công cụ này.

KẾT LUẬN. Nêu ra những điểm đã đạt và chưa đạt được của đồ án, cũng như hướng phát triển trong tương lai cho hướng nghiên cứu này.

# CHƯƠNG I: TỔNG QUAN

## Đặt vấn đề

Một trong những thuộc tính rất quan trọng của hệ thống phần mềm là tính tin cậy[1]. Các kỹ thuật để dự đoán tính tin cậy phần mềm dựa trên các mô hình thiết kế có thể giúp các kiến trúc sư phần mềm đánh giá ảnh hưởng từ thiết kế của họ lên tính tin cậy hệ thống. Điều này có thể giúp hệ thống đáng tin cậy hơn và giảm hoặc tránh các chi phí cho cho việc sửa lỗi. Tuy nhiên, các hướng tiếp cận dự đoán tính tin cậy cho các hệ thống phần mềm hướng thành phần hiện vẫn đang bị giới hạn trong việc áp dụng vào thực tế, nguyên nhân vì hoặc là các mô hình này bỏ qua hoặc không hỗ trợ một cách rõ ràng một số yếu tố ảnh hưởng đến tính tin cậy hệ thống như:

* Lỗi lan truyền
* Cơ chế chịu lỗi phần mềm
* Lỗi đồng thời
  1. Bỏ qua loại lỗi lan tuyền

Lỗi lan truyền là một nhân tố quan trọng có thể dẫn đến lỗi hệ thống, tuy nhiên nhiều hướng dự đoán tính tin cậy hiện nay bỏ qua sự ảnh hưởng của lỗi này. Họ đều giả thuyết rằng lỗi xuất hiện tại một thành phần của hệ thống ngay lập tức ảnh hưởng gây nên lỗi hệ thống, hay nói cách khác là lỗi tại thành phần luôn được lan truyền với xác suất 1.0 lên hệ thống. Trong khi đó, các mô hình hệ thống phần mềm hiện nay không chỉ bao gồm kiểu mô hình thực thi tuyến tính mà còn có nhiều kiểu mô hình khác như mô hình song song hay mô hình chịu lỗi, việc bỏ qua xem xét lỗi lan truyền cho hai ví dụ mô hình vừa nêu trên làm cho các phương pháp tiếp cận này không còn phù hợp để mô hình hóa các hệ thống phần mềm phức tạp.

* 1. Bỏ qua cơ chế chịu lỗi phần mềm

FTMs khá thường được tích hợp trong các hệ thống phần mềm và tạo thành một phần quan trọng để cải thiện tính tin cậy hệ thống. Tuy nhiên, hiện tại nhiều hướng tiếp cận dự đoán tính tin cậy hệ thống không hỗ trợ mô hình hóa FTM

* 1. Bỏ qua các lỗi đồng thời

Trong thực thế, việc xảy ra nhiều lỗi đồng thời là khá phổ biến. Tuy nhiên, theo tìm hiểu và hiểu biết của chúng tôi, các hướng dự đoán tính tin cậy hiện tại không hỗ trợ mô hình hóa các lỗi đồng thời này.

## Mục tiêu và nhiệm vụ đồ án

* Tìm hiểu cách dự đoán tính tin cậy dựa trên cơ sở hướng thành phần.
* Tìm hiểu mô hình Pham-Nordmann-Zhang.
* Tìm hiểu cách áp dụng mô hình Pham-Nordmann-Zhang vào dự đoán tính tin cậy cho các hệ thống phần mềm hướng thành phần.
* Tìm hiểu về cách nâng cao tính tin cậy bằng sử dụng công nghệ cấu trúc chịu lỗi.
* Phát triển công cụ thực tế (Công cụ RMPI - Reliability Modeling, Prediction, and Improvements) với các chức năng sau đây:

- Kiểm tra dữ liệu đầu vào XML có phù hợp với XSD tương ứng được quy định trước hay không.

- Dự đoán được tính tin cậy hệ thống: Đưa ra được con số cụ thể về tính tin cậy.

- Đưa ra được xác suất xuất hiện các loại lỗi của hệ thống.

- Trực tiếp sửa vào file .xml đầu vào để nhìn thấy được sự thay đổi của tính tin cậy. (Trong trường hợp áp dụng công nghệ cấu trúc chịu lỗi đúng, phải đảm bảo tính tin cậy của hệ thống tăng lên)

- Khi mở file .xml đầu vào, hiển thị được toàn bộ thông tin các thành phần, các loại lỗi, cấu trúc của toàn bộ hệ thống để người sử dụng dễ hình dung.

## Định hướng giải quyết vấn đề

Để thực hiện được các mục tiêu và nhiệm vụ đề ra cho đồ án này, cần giải quyết được các vấn đề hiện còn tồn tại trong việc dự đoán tính tin cậy của các hệ thống phần mềm hướng thành phần.

* 1. Xem xét loại lỗi lan truyền

Theo hướng dự đoán tính tin cậy của Cheung [11], một trong những hướng tiếp cận đầu tiên để dự đoán tính tin cậy phần mềm, hướng này biểu diễn luồng điều khiển giữa các module trong phần mềm sử dụng. Một vài hướng tiếp cận khác hiện nay mở rộng hướng tiếp cận của Cheung để hỗ trợ cho nhiều kiểu kiến trúc khác nhau và cũng để kết hợp việc phân tích tính tin cậy và phân tích hiệu năng nhưng đều không xem xét đến các lỗi lan truyền.

Tiếp theo, hướng tiếp cận của Cortellessa [12] giả thiết xác suất thất bại và khả năng gây lỗi lan truyền trên mỗi thành phần là độc lập với nhau. Do đó, xảy ra sự mâu thuẫn khi một bộ phận bị hỏng (lỗi), nó luôn truyền lỗi đó đến bộ phận tiếp. Điều này không phải lúc nào cũng đúng do, một lỗi tính toán bị sai của một thành phần có thể bị viết đè tại đầu vào của thành phần kế tiếp dẫn đến việc đưa ra kết quả đầu ra là chính xác. Hoặc, hướng tiếp cận của Filieri [13] và Mohamed [14] hỗ trợ xử lí nhiều kiểu thất bại khi xem xét lỗi lan truyền. Tuy nhiên, tất cả các hướng tiếp cận nêu trên chỉ xem xét đễn lỗi lan truyền cho mô hình thực thi tuyến tính đơn, bỏ qua việc xem xét lỗi lan truyền song song hay mô hình thực thi chịu lỗi, trong khi đó đây là những mô hình mà hệ thống phần mềm hiện đại hay sử dụng.

* 1. Xem xét về cơ chế chịu lỗi phần mềm (FTMs)

Nhiều hướng dự đoán tính tin cậy không hỗ trợ mô hình cơ chế chịu lỗi phần mềm FTMs. Mặc dù có một vài hướng tiếp cận có đưa FTMs vào mô tả như hướng tiếp cận của Sharma [22], hay hướng tiếp cận của Wang [23]. Tuy nhiên hai hướng tiếp cận này không xem xét đến việc ảnh hưởng của cả việc phát hiện lỗi và xử lý lỗi của FTMs trên luồng điều khiển và luồng dữ liệu của thành phần.

Cụ thể hơn, những hướng tiếp cận này giả thiết rằng khi một lỗi thuộc một loại thất bại cố định gây ra thất bại của thành phần, FTMs luôn có thể xử lý thất bại đó nếu FTMs được đặt ra với mục đích là xử lý được loại thất bại đó. Có nghĩa rằng FTM có thể xử lý lỗi một cách hoàn hảo của loại thất bại xác định đó (xác suất để phát hiện được thất bại đó bằng 1).

Tuy nhiên, trong thực thế, xác xuất để xác định được lỗi là không hoàn hảo nên FTMs có thể để lỗi gây ra bởi một thành phần lan truyền mà không kích hoạt việc xử lý lỗi. Việc bỏ qua ảnh hưởng của cả phát hiện lỗi và xử lý lỗi trong luồng điều khiển và luồng dữ liệu của FTMs trên mỗi bộ phận có thể dẫn đến dự đoán sai lệch kết quả khi các hoạt động của FTMs bị sai lệch so với các đặc tả của người tạo ra.

* 1. Xem xét lỗi đồng thời

Trong phạm vi tìm hiểu của tôi, hiện tại các hướng tiếp cận đều chưa hỗ trợ việc xem xét lỗi đồng thời. Nói cách khác, các hướng tiếp cận đó chỉ hỗ trợ một lỗi tại một thời điểm.

## Kết chương

Nhiều hướng tiếp cận được nêu đến ở trên, nhưng nếu xét thổng thể về 3 vấn đề: lỗi lan truyền, cơ chế chịu lỗi phần mềm FTMs và lỗi đồng thời thì hấu hết các hướng tiếp cận đều chưa xử lý được đầy đủ.

Hướng tiếp cận trong đồ án này đề cập và phát triển mở rộng các vấn đề:

- Xem xét một cách tổng thể sự kết hợp lỗi lan truyền trong các trường hợp tuyến tính, song song, và mô hình thực thi chịu lỗi.

- Đề cập một cách rõ ràng các khía cạnh hoạt động liên quan đến tính tin cậy thích đáng của FTMs.

- Xem xét đến lỗi đồng thời.

🡪 Sau khi giải quyết được các vấn đề còn tồn tại cho việc dự đoán tính tin cậy các hệ thống phần mềm hướng thành phần, trong phạm vi đồ án này, tôi phát triển một công cụ bằng Java để ám dụng các tìm hiểu trên lý thuyết đã tìm hiểu vào thực tế.

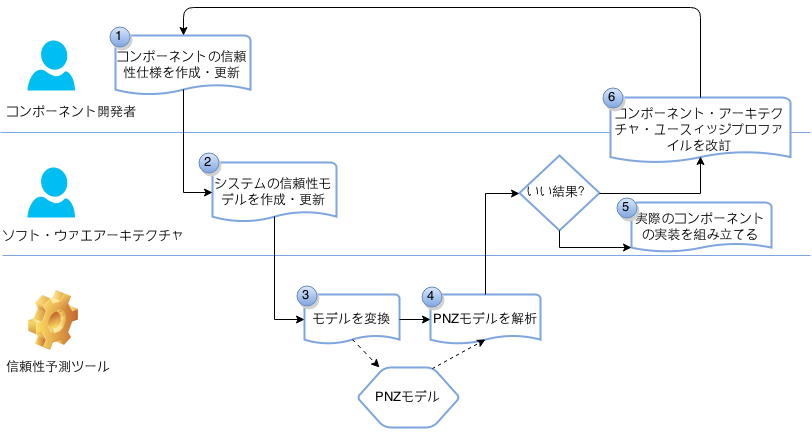
# CHƯƠNG II: DỰ ĐOÁN ĐỘ TIN CẬY VÀ MÔ HÌNH PNZ

## Dự đoán tính tin cậy dựa trên cơ sở hướng thành phần

Hướng thành phần là chia một hệ thống phần mềm thành nhiều thành phần có chứa những nội dung và biểu hiện có thể thay thế được. Một thành phần có những hành vi riêng của nó được định nghĩa để cung cấp và yêu cầu các giao diện tương ứng. Tuy nhiên, để dự đoán tính tin cậy của kiến trúc phần mềm dựa trên cơ sở hướng thành phần, chúng ta cần thêm các thông tin bổ sung khác nữa về mỗi thành phần.

Do có một sự phân biệt khá rõ ràng giữa người phát triển thành phần và kiến trúc sư phần mềm trong Kỹ thuật phần mềm dựa trên cơ sở hướng thành phần (CBSE), do đó khi tạo các đặc tả (hoặc mô hình) để lấy thông tin bổ sung cần chú ý đến hai vai trò này. Khi đó, người phát triển thành phần thực thi thành phần, và không chỉ cung cấp thành phần đặc tả chức năng mà còn cung cấp cả thành phần đặc tả tính tin cậy. Kiến trúc sư phần mềm sử dụng các thành phần đặc tả tính tin cậy này và cung cấp thêm các hồ sơ sử dụng (usage profile) của hệ thống để dự đoán tính tin cậy.

Một thành phần đặc tả tính tin cậy cần mô tả các hành vi của dịch vụ được cung cấp bởi thành phần. Ví dụ: làm sao để các dịch vụ được cung cấp đáp ứng cho các dịch vụ được yêu cầu… Từ đó, bằng việc thu thập lại các đặc tả và cung cấp thêm hồ siw sử dụng của hệ thống, các kiến trúc sư phần mềm tạo các mô hình tính tin cậy hệ thống. Trong phần 3 của chương này, theo tìm hiểu, tôi cung cấp giản đồ mô hình tính tin cậy hỗ trợ người phát triển thành phần tạo thành phần đặc tả tính tin cậy và hỗ trợ kiến trúc sư phần mềm tạo mô hình tính tin cậy hệ thống.



Hình 1. Hướng dự đoán tính tin cậy dựa trên cơ sở hướng thành phần

Hướng tiếp cận mà tôi tìm hiểu đi theo 6 bước [7] theo như hình trên.

* + - Bước 1: Những người phát triển thành phần cung cấp thành phần đặc tả tính tin cậy. Một thành phần đặc tả tính tin cậy bao gồm các xác xuất tính tin cậy (xác suất thất bại, xác suất lỗi lan truyền) và các dịch vụ đầu vào, đầu ra tương ứng cho mỗi thành phần. Làm cách nào để xác định các xác suất này nằm ngoài phạm vi nghiên cứu của tôi trong giai đoạn này.
    - Bước 2: Kiến trúc sư phần mềm tạo mô hình tính tin cậy hệ thống bằng việc thu thập thành phần đặc tả tính tin cậy theo kiến trúc hệ thống đã được thiết kế và cung cấp thêm hồ sơ sử dụng của hệ thống để hoàn thiện. (Ví dụ: các tác động trực tiếp đến người sử dụng hoặc các hệ thống khác.)
    - Bước 3: Từ mô hình tính tin cậy hệ thống chuyển thể thành mô hình Pham-Nordmann-Zhang (PNZ).
    - Bước 4: Bằng việc phân tích mô hình PNZ, dự đoán tính tin cậy cho mỗi dịch vụ được cung cấp có thể được thu thập dựa vào các xác xuất tính tin cậy của thành phần trong kiến trúc hệ thống. Để hỗ trợ bước 3 và 4 tôi cung cấp một công cụ dự đoán tính tin cậy mà việc chuyển đổi dự đoán tính tin cậy được mô tả một cách rõ ràng trong chương III.
    - Nếu kết quả dự đoán chỉ ra rằng yêu cầu tính tin cậy không được đáp ứng thì bước 5 được thực hiện, nếu không bước 6 được thực hiện.
    - Bước 5: Có nhiều lựa chọn để thực hiện:

+ Người phát triển thành phần có thể xét duyệt lại thành phần đó [16]. Như: thay đổi định dạng của FTMs…

+ Kiến trúc sư phần mềm có thể xét duyệt lại kiến trúc phần mềm và hồ sơ sử dụng của hệ thống [17]. Như: thử định dạng khác của kiến trúc hệ thống, hay thay thế một số thành phần của hồ sơ sử dụng hệ thống…

* + - Bước 6: Hệ thống được mô hình đã đáp ứng các yêu cầu tin cậy đề ra và kiến trúc sư hệ thống thu thập bộ phận thực thi thực tế theo đúng kiến trúc hệ thống.

## Mô hình Pham-Nordmann-Zhang (PNZ)

Mô hình PNZ là một dạng thuộc nhóm các mô hình của quá trình Poisson không đồng nhất [8].

Hãy sử dụng một số chức năng để mô tả đặc điểm của hệ thống khi nó được mô hình hóa bằng mô hình của quá trình Poisson không đồng nhất trong Bảng 1.

|  |  |
| --- | --- |
| a(t) | Tổng số lỗi |
| b(t) | Tỉ lệ phát hiện lỗi |
| m(t) | Số lỗi dự kiến sẽ phát hiện trong thời gian *t* |
| (t) | Tần suất lỗi |

Bảng 1. Các đặc tả sử dụng trong quá trình Poisson không đồng nhất

Tại thời điểm *t*, hệ thống có a(t) lỗi và m(t) lỗi đã được phát hiện nên ta còn a(t)-m(t) lỗi. Với tỉ lệ phát hiện lỗi là b(t), chúng ta có mỗi quan hệ giữa các lỗi được phát hiện trong khoảng thời gian t, số lỗi còn lại của hệ thống, và tỉ lệ phát hiện lỗi như sau:

Trong đó, là giá trị vô cùng nhỏ với : . Khi , ta có:

Nếu gọi t0 là thời điểm bắt đầu của quá trình kiểm thử, với điều kiện bạn đầu: m và . Nhà khoa hoc Pham đã chứng minh, kết quả của biểu thức trên là:

Trong đó:

TS. Phạm Hoàng [15] đã giới thiệu mô hình tính tin cậy hệ thống thuộc nhóm các mô hình của quá trình Poisson không đồng nhất với các thuộc tính phụ thuộc thời gian như sau:

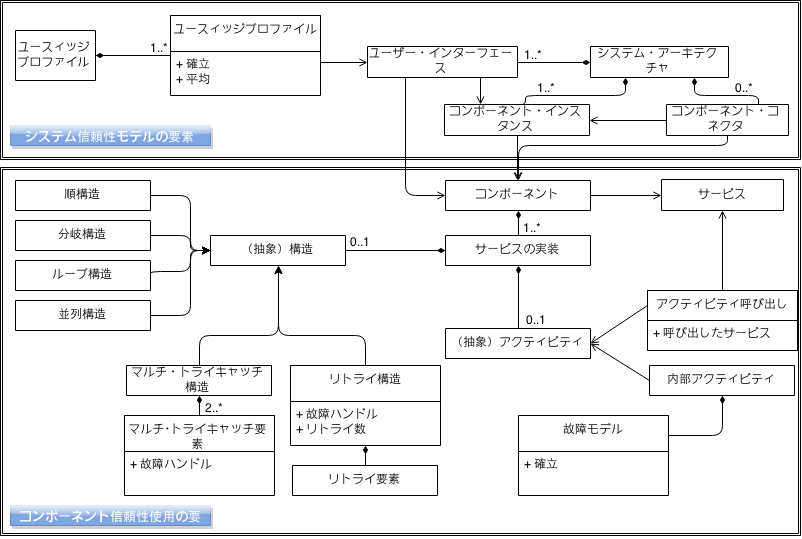
Khi đó:

Theo công bố của nhiều ấn phẩm, mô hình PNZ là một trong những mô hình tốt nhất trong nhóm các mô hình của quá trình Poisson không đồng nhất.

## Mô hình hóa tính tin cậy

Trong phần này, theo các kiến thức đã tìm hiểu, tôi mô tả giản đồ mô hình hóa tính tin cậy giúp hỗ trợ người phát triển thành phần tạo bộ phận đặc tả tính tin cậy và kiến trúc sư phần mềm tạo mô hình tính tin cậy phần mềm.

* 1. Bộ phận đặc tả tính tin cậy
     1. Các dịch vụ, các thành phần, các thực thi dịch vụ



Hình 2. Sơ đồ các bộ phận đặc tả tính tin cậy

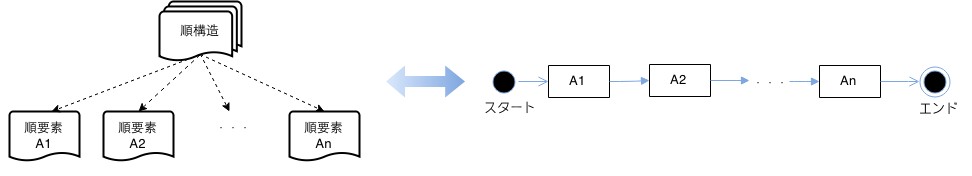
Hình trên đưa ra thông tin các thành phần cần thiết cho việc lập Tài liệu đặc tả tính tin cậy hệ thống của người phát triển thành phần và Mô hình tính tin cậy hệ thống của kiến trúc sư phần mềm.

Người phát triển thành phần mô hình các thành phần, các dịch vụ, và các thực thi dịch vụ thông qua các thành phần mô hình hóa: *bộ phận, dịch vụ và thực thi dịch vụ* một cách tương ứng. Các thành phần liên lạc với các dịch vụ thông qua *Yêu cầu dịch vụ và cung cấp dịch vụ*.

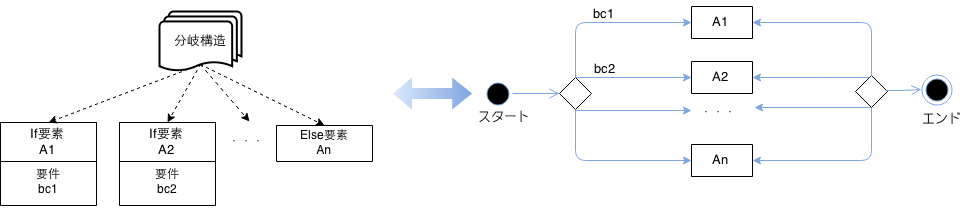
Có 2 loại hoạt động:

- Hoạt động nội bộ: đại diện cho các tính toán bên trong của thành phần.

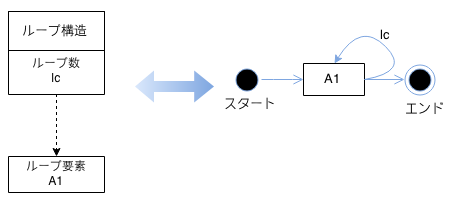
- Hoạt động gọi: đại diện cho một gọi đồng bộ đến các thành phần khác, nó chính là một khối gọi cho đến khi nhận được hồi đáp. Dịch vụ được gọi của hoạt động là một trong các dịch vụ được cung cấp của thành phần hiện tại, sự xem xét yêu cầu dịch vụ này chỉ có thể được thay thế bởi dịch vụ được cung cấp của thành phần khác khi liên kết của thành phần hiện tại tới các thành phần khác là cố định.



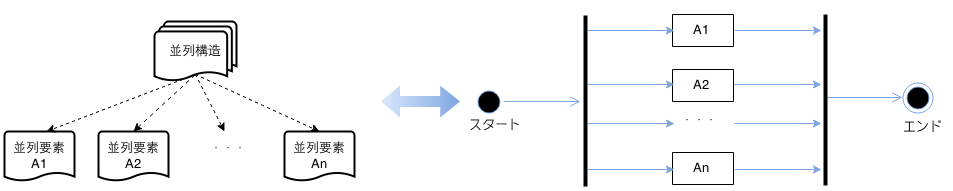
Hình 3. Cấu trúc tuyến tính



Hình 4. Cấu trúc rẽ nhánh



Hình 5. Cấu trúc vòng lặp



Hình 6. Cấu trúc song song

Có 4 loại cấu trúc luồng điều khiển cơ sở được cung cấp trong giản đồ mô hình hóa tính tin cậy phần mềm mà tôi tìm hiểu.

- Cấu trúc tuyến tính (*cấu trúc tuyến tính*), các thành phần tuyến tính (*thành phần tuyến tính*) được thực thi tuần tự.

- Cấu trúc rẽ nhánh (*cấu trúc rẽ nhánh*), thừa kế tính chất của cấu trúc tuyến tính, chỉ khác là việc lựa chọn bước tiếp theo phụ thuộc vào điều kiện rẽ nhánh.

- Cấu trúc vòng lặp (*cấu trúc vòng lặp*), có một bộ phận vòng lặp đơn. Không xét đến trường hợp lặp vô hạn.

- Cấu truc song song (*cấu trúc song song*), thường được sử dụng trong môi trường thực thi đồng thời nơi các bộ phận đơn thực thi đồng thời để tăng hiệu năng sự dụng cho hệ thống.

* + 1. Các mô hình thất bại

Trong phần này chúng ta sẽ định nghĩa các khái niệm “error”, “fault”, “failure” và mối quan hệ giữa chúng ở mức độ trừu tượng hóa nhằm mục đích xác định được định nghĩa các mô hình thất bại dành cho các hoạt động nội bộ.

Avizienis [9] đã đưa ra định nghĩa: error là một phần của toàn bộ các trạng thái của hệ thống có khả năng dẫn đến failure. Nguyên nhân của error gọi là fault. Một failure xuất hiện khi error có nguyên nhân do dịch vụ phân phát đi sai lệch so với dịch vụ chính xác.

* Error có thể xuất hiện bởi lỗi nội bộ. Ví dụ: việc sai sót trong thực thi mã nguồn của hoạt động nội bộ của thành phần là một lỗi nội bộ. Lỗi này dẫn đến error trong trạng thái nội bộ nếu mã nguồn này được thực thi.
* Error cũng có thể xuất hiện bởi lỗi ngoài. Ví dụ: một sai sót khi nhập đầu vào sẽ xuất hiện lỗi và lan truyền lỗi đó đến các thành phần thông qua giao diện của nó.
* Error cũng có thể xuất hiện bởi cả lỗi nội bộ và lỗi ngoài.

Tuy nhiên, không phải mọi error đều dẫn đến failure trong thành phần. Một failure trong thành phần chỉ diễn ra khi một error trong thành phần đó gây ra lan truyền. Tương tự, không phải mọi failure trong thành phần đều dẫn đến failure hệ thống.

Trong quá trình lan truyền:

* Error có thể bị phát hiện và do đó ngăn chặn lan truyền.

Ví dụ: một lỗi nhập đầu vào bị phát hiện bởi hệ thống phát hiện lỗi của hoạt động nội bộ.

* Một lỗi có thể bị đánh dấu.

Ví dụ: một giá trị lỗi bị viết đè bởi việc tính toán của hoạt động nội bộ trước khi di chuyển qua giao diện.

* Một lỗi có thể bị chuyển đổi.

Ví dụ: lỗi thời gian nhận từ các thành phần dịch vụ khác có thể dẫn đến thành phần dịch vụ hiện tại thực hiện tính toán với dữ liệu lỗi thời làm xảy ra lỗi nội dung. Một lỗi có thể xuất hiện đòng thời với một lỗi khác.

Người phát triển thành phần mô hình các loại failure các khau (*Loại lỗi*) bằng việc sử dụng cây cấp bậc các loại lỗi. Ngoại trừ loại thất bại F0– một loại thất bại tiền định nghĩa tương ứng với dịch vụ chính xác, người phát triển thành phần mô hình một loại lỗi bằng việc mở rộng hoặc là *Loại lỗi dừng* hoặc là *Loại lỗi lan truyền*. Các loại thất bại mở rộng của *Loại lỗi dừng* có liễn quan đến các error có thể được xác định và được đánh dấu với một ký hiệu cảnh báo bởi hệ thống xác định lỗi của hoạt động nội bộ. Trong khi đó,các loại thất bại mở rộng của *Loại lỗi lan truyền* liên quan đến các error không thể xác định và đánh dấu bởi hệ thống xác định lỗi của hoạt động nội bộ.

Định nghĩa 1:

- F0 là loại thất bại tiền định nghĩa tương ứng với dịch vụ chính xác.

- FS là nhóm tất cả các lỗi dừng {FS1, FS2, …, FSu}.

- FP là nhóm tất cả các lỗi lan truyền {FP1, FP2, …, FPv}.

- AIOS lá nhóm của tất cả các nhóm thất bại cho hoạt động nội bộ có đầu vào hoặc đầu ra là {{F0}} (2Fp\🛇)



- AFS là nhóm của tất cả các nhóm loại thất bại cho hoạt động nội bộ với dấu hiệu thất bại {{FS1},…,{FSu}}.

* + 1. Các cấu trúc chịu lỗi

Trong các nghiên cứu trước đây, Avizienis mô tả cụ thể luật của FTMs. Một FTM được thực thi thông qua xác định lỗi và hệ thống khôi phục.

* Xác định lỗi là định danh sự tồn tại của lỗi hiện thời.
* Xử lý lỗi là loại bỏ lỗi khỏi trạng thái hệ thống.

Ví dụ: Đưa hệ thống trở lại trạng thái đã được lưu trữ trước đó🡪 xử lý lỗi là việc ngăn chặn lỗi xuất hiện một lần nữa.

Để hỗ trợ mô hình FTMs, giản đồ mô hình hóa tính tin cậy mà tôi tìm hiểu cung cấp các cấu trúc chịu lỗi (FTSs), cụ thể là *Cấu trúc thử lại* và *Cấu trúc đa loại trừ*. Đối với FTMs, xác định lỗi là điều kiện tiên quyết cho xử lý lỗi và không phải tất cả lỗi được xác định đều xử lý được. Do đó, tối đa một *Cấu trúc thử lại* hoặc một *Cấu trúc đa loại trừ* có thể cung cấp xử lý lỗi chỉ cho các thất bại đã được đánh dấu, chính là kết quả của các error có thể được xác định và đánh dấu [10].

* 1. Mô hình hóa tính tin cậy hệ thống

Trong hướng tiếp cận mà tôi tìm hiểu, các kiến trúc sư phần mềm thu được các thành phần và đặc tả tính tin cậy của chúng từ người phát triển thành phần. Sau đó, họ cung cấp hồ sơ sử dụng của hệ thống cho hệ thống hoàn thiện để định hình mô hình tính tin cậy hệ thống.

Sau khi mô hình hóa kiến trúc hệ thống, các kiến trúc sư phần mềm mô hình hồ sơ sử dụng của hệ thống cho giao diện người dùng. Một hồ sơ sử dụng của hệ thống chứa các thành phần hồ sơ sử dụng với các thuộc tính khác nhau, cũng như mô hình các kịch bản hồ sơ sử dụng khác nhau của hệ thống. Một thành phần hồ sơ sử dụng của hệ thống phải chứa đầy đủ thông tin để quyết định các xác suất phân nhánh của cấu trúc phân nhánh và con số trung bình các vòng lặp cho mỗi cấu trúc vòng lặp.

## Dự đoán tính tin cậy ứng dụng mô hình PNZ

Sau khi các kiến trúc sư phần mềm đã thu thập thành phần đặc tả tính tin cậy để nhận yêu cầu chức năng và đặc tả hồ sơ sử dụng của hệ thống để định hình mô hình tính tin cậy hệ thống, chúng ta có thể dự đoán tính tin cậy của toàn bộ hệ thống hoàn thiện. Tiến trình dự đoán tính tin cậy hệ thống bắt đầu với mô hình tính tin cậy hệ thống và thành phần đặc tả tính tin cậy, và kết thúc với đầu ra là dự đoán tính tin cậy. Nó bao gồm sự chuyển đổi mỗi thành phần cách sử dụng hệ thống và một tập hợp các kết quả.

### Chuyển đổi mỗi thành phần “UsageProfile”

Sự chuyển đổi bắt đầu với việc thực thi dịch vụ của chính dịch vụ được cung cấp. Theo thiết kế, giản đồ chuyển đổi bao gồm:

1. Một thực thi dịch vụ có thể bao gồm một cấu trúc chứa nhiều loại cấu trúc và một hoạt động chứa nhiều loại hoạt động.
2. Một thành phần bên trong một cấu trúc (Ví dụ: *Thành phần tuyến tính, thành phần nếu, thành phần thì, thành phần vòng lặp, thành phần song song, thành phần thử lại, thành phần đa loại trừ*) có thể chứa một cấu trúc khác, hoặc một hoạt động khác.
3. Một lời gọi hoạt động thường thì sẽ tham khảo đến một thực thi dịch vụ khác. Do đó, sự biến đổi về cơ bản là một thủ tục đệ quy áp dụng cho các cấu trúc.

Trong mỗi cấu trúc, sự chuyển đổi biến đổi cấu trúc đó vào một hoạt động nội bộ tương đương. (Gọi tắt là IA tương đương)

#### Cấu trúc tuyến tính

Xem xét một cấu trúc tuyến tính với n thành phần tuyến tính A1, A2, …, An Lấy PrA12…k(I,FO), IAIOS, FO(AFS AIOS) là mô hình thất bại cho IA tương đương của k thành phần tuyến tính đầu tiên, khi đó mô hình thất bại cho IA tương đương của k+1 thành phần tuyến tính đâu tiên sẽ được tính toán như dưới đây.



* k+1 thành phần tuyến tính đầu tiên sẽ đưa ra kết quả đúng nếu k thành phần tuyến tính đầu tiên tạo ra một kết quả (đúng hay sai) và sau đó nhận kết quả đó làm đầu vào, thành phần tuyến tính thứ k+1 đưa ra kết quả đúng:
* k+1 thành phần tuyến tính đầu tiên đưa ra một lỗi thuộc loại lỗi dừng F (FAFS) nếu: hoặc là k thành phần tuyến tính đầu tiên đưa ra lỗi thuộc loại lỗi dừng hoặc là k thành phần tuyến tính đầu tiên đưa ra một kết quả (đúng hay sai) và sau đó nhận kết quả đó làm đầu vào, thành phần tuyến tính thứ k+1 đưa ra lỗi thuộc loại lỗi dừng F:



* k+1 thành phần tuyến tính đầu tiên đưa ra một lỗi thuộc loại lỗi lan truyền OAIOS\{{F0}} k thành phần tuyến tính đầu tiên đưa ra một kết quả (đúng hay sai) và sau đó nhận kết quả đó làm đầu vào, thành phần tuyến tính thứ k+1 đưa ra lỗi thuộc loại lỗi lan truyền O:



Bằng việc sử dụng 3 công thức ở trên việc chuyển đổi đệ quy tính toán ra mô hình thất bại cho IA tương đương với tất cả n thành phần tuyến tính được tiến hành như sau: (ví dụ: mô hình thất bại cho IA tương đương của cấu trúc tuyến tính):

#### Cấu trúc rẽ nhánh

Xem xét một cấu trúc rẽ nhánh với n-1 “thành phần nếu” A1, A2, …,An-1 và một “thành phần thì” duy nhất An, khi đó IA tương đương có mô hình thất bại như sau (với IAIOS, FO(AFS AIOS)):



Với p(bci)(Với i=1,2,…n-1) là xác suất của điều kiện rẽ nhánh bci (ví dụ: khả năng thực hiện nhánh nếu Ai) thu được từ thành phần cách sử dụng hệ thống hiện tại.

#### Cấu trúc vòng lặp

Xem xét một cấu trúc vòng lặp với một vòng lặp đơn A1, cấu trúc vòng lặp này có thể chuyển đổi thành cầu trúc tuyến tính với lc cấu trúc thành phần tuyến tính A1. Khi đó, với trung bình cộng các vòng lặp, averge(lc), chứa thành phần hồ sơ sử dụng của hệ thống hiện tại, mô hình thất bại cho IA tương đương của cấu trúc vòng lặp có thể được tính toàn bằng việc sử dụng phương thức chuyển đổi tương tự như đối với áp dụng cho cấu trúc tuyến tính, cho cấu trúc tuyến tính tương đương đã được chuyển đổi từ cấu trúc vòng lặp. Tuy nhiên, do tất cả các thành phần tuyến tính của của cấu trúc tuyến tính tương đương sau khi chuyển đổi đều có cùng A1, việc chuyển đổi đồng thời sử dụng các lũy thừa để chuyển đổi nhanh hơn.

#### Cấu trúc song song

Đối với cấu trúc song song, phương thức chuyển đổi sẽ chuyển đổi cấu trúc này thành các IA tương đương dựa trên các lập luận sau đây:

* Cấu trúc song song (theo đó là IA tương đương) đưa ra kết quả thất bại nếu ít nhất một nhánh song song đưa ra kết quả thất bại.
* Cấu trúc song song (theo đó là IA tương đương) đưa ra kết quả chính xác nếu tất cả các nhánh song song đưa ra kết quả chính xác.
* Cấu trúc song song (theo đó là IA tương đương) đưa ra kết quả sai (erroneous) nếu không có nhánh song song nào gặp lỗi và ít nhất một nhánh song song đưa ra kết quả sai.

Và ta có được kết luận sau đây:

* Các hành vi có liên quan đến tính tin cậy của các nhánh song song là độc lập.
* Trong trường hợp cấu trúc song song nhận được đầu vào lỗi thuộc loại lỗi lan truyền, mỗi nhánh song song của nó nhận đầu vào là lỗi thuộc cùng loại lỗi lan truyền đó. Và trong trường hợp cấu trúc song song đưa ra đầu ra lỗi, loại lỗi lan truyền của lỗi đầu ra của cấu trúc song song là hợp của các loại lỗi lan truyền của lỗi đầu ra của mỗi nhánh trong cấu trúc.
* Khi các nhánh song song đưa ra kết quả thất bại thuộc loại lỗi dừng, lỗi dừng của toàn bộ cấu trúc song song sẽ là lỗi dừng của nhánh song song có chỉ số thấp nhất.

Xem xét một cấu trúc song song với n nhánh song song A1, A2,…,An, lấy PrA12…k(I,FO), IAIOS, FO(AFS AIOS) là mô hình thất bại cho IA tương đương của k nhánh song song đầu tiên, khi đó mô hình thất bại cho k+1 nhánh song song đầu tiên sẽ được tính như sau:



* k+1 nhánh song song đầu tiên đưa ra kết quả chính xác khi k nhánh song song đầu tiên đưa ra kết quả chính xác và nhánh song song thứ k+1 cũng có kết quả chính xác:
* k+1 nhánh song song đầu tiên đưa ra thất bại thuộc loại lỗi dừng F (với FAFS) khi hoặc là k nhánh song song đầu tiên đưa ra thất bại thuộc loại lỗi dừng F hoặc là k nhánh song song đầu tiên đưa ra kết quả (đúng hoặc sai) và nhánh song song thứ k+1 đưa ra thất bại thuộc loại lỗi dừng F:



* k+1 nhánh song song đầu tiên đưa ra lỗi thuộc loại lỗi lan truyền OAIOS\{{F0}} nếu: hoặc là k nhánh song song đầu tiên đưa ra kết quả chính xác và nhánh song song thứ k+1 đưa ra lỗi thuộc loại lỗi lan truyền O; hoặc là k nhánh song song đầu tiên đưa ra lỗi thuộc loại lỗi lan truyền O và nhánh song song thứ k+1 đưa ra kết quả chính xác; hoặc là k nhánh song song đầu tiên đưa ra lỗi thuộc loại lỗi lan truyền O1AIOS\{{F0}} và nhánh song song thứ k+1 đưa ra đưa ra lỗi thuộc loại lỗi lan truyền O2AIOS\{{F0}} sao cho :



Bằng việc sử dụng 3 công thức phía trên việc chuyển đổi đệ quy sẽ tính toán mô hình thất bại cho IA tương đương của toàn bộ n nhánh song song (ví dụ: mô hình thất bại dành cho IA tương đương của cấu trúc song song):

#### Cấu trúc thử lại

Xem xét một cấu trúc thử lại, lấy rc là biến đếm số lần thử lại, là tập các thất bại được xử lý, với tất cả IAIOS, FO(AFS AIOS) là mô hình thất bại cho IA tương đương của *thành phần thử lại* (viết tắt là RP).



Theo tìm hiểu của tôi, với mỗi đầu vào IAIOS của *cấu trúc thử lại*, quá trình chuyển đổi xây dựng một mô hình Markov phản ánh tất cả các quá trình thực thi có thể của *cấu trúc thử lại* với đầu vào I và các xác suất tương ứng của nó, rồi sau đó xây dựng mô hình thất bại cho IA tương đương từ mô hình Markov.



Tuy nhiên, cụ thể trong đồ án này tôi sẽ ứng dụng mô hình PNZ thay thế cho mô hình Markov. Mô tả cụ thể được nêu trong chương III, mục 2.1.

#### Cấu trúc đa loại trừ

Xem xét một cấu trúc đa loại trừ, lấy n là số *thành phần loại trừ*, với mỗi , gọi là tập các thất bại đã được xử lý của *thành phần loại trừ* thứ i, với tất cả IAIOS, FO(AFS AIOS) là mô hình thất bại cho *thành phần loại trừ* thứ i (viết tắt là MPi).



Cũng giống như *cấu trúc thử lại,* theo tìm hiểu của tôi,cho mỗi cho mỗi đầu vào có thể của *cấu trúc đa loại trừ*, quá trình chuyển đổi xây dựng một mô hình Markov phản ánh tất cả các quá trình thực thi có thể của *cấu trúc đa loại trừ* với đầu vào I và các xác suất tương ứng, sau đó xây dựng mô hình thất bại cho IA tương đương từ mô hình Markov.

Tuy nhiên, cụ thể trong đồ án này tôi sẽ ứng dụng mô hình PNZ thay thế cho mô hình Markov. Mô tả cụ thể được nêu trong Chương III, mục 2.1.

### Tổng kết các kết quả để đưa ra tính tin cậy hệ thống

Kết quả của tính tin cậy của dịch vụ được cung cấp được tổng kết như sau: gọi R(UPPj) là xác suất của dịch vụ được cung cấp mà cách sử dụng hệ thống thành phần UPPj trỏ tới, m là số thành phần cách sử dụng hệ thống có, Pj là xác suất mà người sử dụng truy cập với hồ sơ sử dụng của hệ thống thành phần UPPj như khi đó tính tin cậy tổng quát của toàn bộ hệ thống có thể được tính như sau:

### Cách thực thi

Tôi đã thực hiện các thuật toán chuyển đổi trong công cụ dự đoán độ tin cậy mà tôi tìm hiểu và phát triển. Công cụ này sẽ nhận mô hình Pham-Nordmann-Zhang làm một đầu vào, xác nhận đầu vào này đối với một tập các ràng buộc ngữ nghĩa theo giản đồ mô hình độ tin cậy của tôi (ví dụ như tổng xác suất của tất cả các cách sử dụng hệ thống phải có 1), và đưa ra đầu ra là hệ thống dự đoán tính tin cậy. Kết quả này không chỉ bao gồm dự đoán độ tin cậy hệ thống mà còn bao gồm xác suất thất bại và loại thất bại.

Tôi cũng thực hiện một mô phỏng độ tin cậy bằng công cụ của tôi. Công cụ nhận mô hình Pham-Nordmann-Zhang làm đầu vào. Nó có khả năng để kiểm soát việc thực hiện của từng hoạt động nội bộ theo mô hình thất bại của nó. Để mô phỏng các mô hình thất bại đối với một hoạt động nội bộ, tôi thực hiện một phương pháp như sau: (1) phương pháp nhận một đầu vào,và trả đầu ra như một thể ném ngoại lệ, (2) Nếu phương pháp này nhận được một đầu vào , nó ném ra một ngoại lệ với xác suất hoặc trả về một kết quả được đánh dấu với xác suất . Tôi cũng thực hiện một phương pháp để mô phỏng từng dịch vụ cung cấp của một thành phần. Phương pháp này cũng nhận một đầu vào, và trả về đầu ra như một thể ném ngoại lệ. Nội dung của phương pháp này bao gồm các báo cáo luồng các dữ liệu dựa theo thực hiện các dịch vụ cung cấp và cung cấp cách sử dụng hệ thống. Cuối cùng, mô phỏng xác định độ tin cậy của hệ thống như tỷ lệ các dịch vụ thực hiện thành công (bắt đầu từ đầu vào chính xác {F0} và kết thúc với kết quả đầu ra chính xác {F0}) để tính tổng thể thực hiện.

So với các phương pháp phân tích của tôi, mô phỏng là còn khá chậm và không thể được sử dụng như phương pháp dự đoán chính. Tuy nhiên, chúng ta có thể sử dụng nó với mục đích là xác nhận tính đúng đắn của lý thuyết. Bằng cách so sánh kết quả dự đoán thu được bằng phương pháp phân tích của tôi với các mô phỏng của hệ thống, nó có thể cho tôi cung cấp bằng chứng cho tính đúng đắn của thuật toán chuyển đổi và tính hợp lệ của kết quả dự đoán.

## Nâng cao tính tin cậy hệ thống phần mềm bằng cấu trúc chịu lỗi

Trong tài liệu tham khảo, nhà khoa học Avizvienis đã miêu tả một cách đầy đủ tất cả các quy tắc của cơ chế cấu trúc chịu lỗi (FTMs). Một FTM được thể hiện thông qua các phát hiện lỗi và xử lý lỗi. Việc xử lý lỗi đồng nghĩa với việc loại bỏ lỗi đó ra khỏi hệ thống. Ví dụ: đưa hệ thống trở lại trang thái an toàn đã được lưu lại trước khi lỗi xuất hiện. Ngoài ra còn cần xử lý nguyên nhân gây ra lỗi. Để xử lý nguyên nhân này, chúng ta có thể: chuyển đổi sang một thành phần dự phòng đang không ở trạng thái lỗi.

Để hỗ trợ mô hình hóa FTMs, mô hình mô hình hóa tính tin cậy mà tôi tìm hiểu cung cấp cấu trúc chịu lỗi (FTSs) [18][19][20][21], cụ thể là cấu trúc thử lại và cấu trúc đa loại trừ. Theo tôi tìm hiểu thì TS. Phạm Thành Trung [7] đã giới thiệu một số khái niệm cơ bản về FTMs. Trong FTM, việc phát hiện lỗi là một điều kiện tiên quyết để có thể xử lý lỗi, tuy nhiên không phải mọi lỗi được phát hiện đều có thể xử lý được. Do đó, hầu hết, một cấu trúc thử lại và cấu trúc đa loại trừ chỉ có thể cung cấp việc xử lý lỗi cho các loại lỗi dừng.

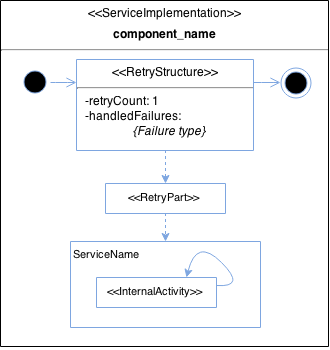
Với cấu trúc thử lại: Kỹ thuật tái thực hiện được sử dụng để xử lý lỗi. Cấu trúc này bao gồm nhiều thành phần thử lại, mà mỗi thành phần này có thể chứa nhiều loại hoạt động, cáu trúc khác nhau, hay thậm chí là một cấu trúc thử lại con. Về cơ bản, khi thành phần thử lại đầu tiên mô hình hóa các dịch vụ bình thường thì thành phần thử lại tiếp theo sẽ mô hìn hóa lại các dịch vụ trước đó.

Với cấu trúc đa loại trừ: cấu trúc đa loại trừ lấy ý tưởng xử lý ngoại lệ trong lập trình hướng đối tượng. Cấu trúc này bao gồm bằng hoặc hơn hai thành phần đa loại trừ. Mỗi thành phần đa loại trừ có thể chứa nhiều loại hoạt động, cáu trúc khác nhau, hay thậm chí là một cấu trúc đa loại trừ con. Giống như các khối loại trừ trong xử lý ngoại lệ, khi thành phần đa loại trừ đầu tiên mô hình hóa các dịch vụ bình thường thì thành phần đa loại trừ tiếp theo xử lý các thất bại nhất định của lỗi dừng và chạy các hoạt động thay thế.

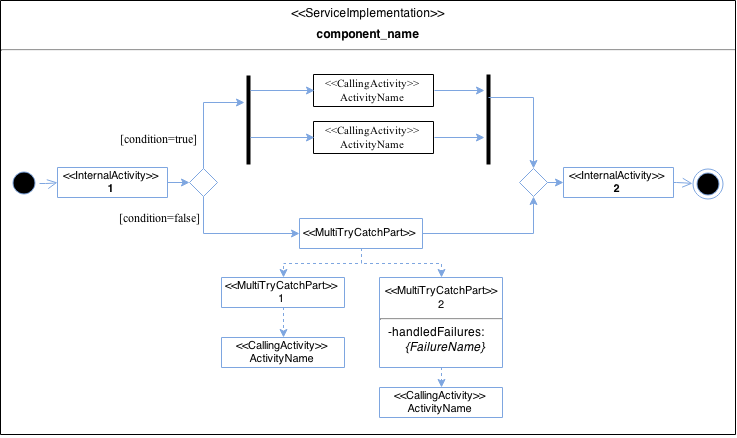
Sau khi có được kết quả tính tin cậy hệ thống, cũng như tỉ lệ xuất hiện các lỗi bằng sử dụng công cụ RMPI. Chúng ta sẽ xác định loại lỗi nào có tỉ lệ xuất hiện cao nhất là làm theo 3 bước dưới đây để làm giả khả năng xuất hiện lỗi đó và từ đó nâng cao tính tin cậy chung của toàn bộ hệ thống

Trong mô hình mà kiến trúc sư hệ thống đã thiết kế từ ban đầu, chúng ta sẽ thêm vào một module gọi là module chịu lỗi, sẽ nằm giữa 2 module đã được xây dựng và là trước module có chứa lỗi có tỉ lệ xuất hiện cao nhất đã được xác định ở trên.

Module chịu lỗi có câu trúc như sau:



Hình 7. Thành phần cấu trúc chịu lỗi sử dụng cấu trúc thử lại



Hình 8. Thành phần cấu trúc chịu lỗi sử dụng cấu trúc đa loại trừ

Chúng ta sẽ đi theo 3 bước:

Bước 1: Mô hình module chịu lỗi theo cấu trúc như trên

Bước 2: Tạo instance cho module

Bước 3: Sửa lại kết nối đến các module

Quá trình chạy thực tế được viết cụ thể trong Chương III, mục 3.2.

## Kết chương

Trong chương này, chúng ta đã nắm được 6 bước để dự đoán tính tin cậy dựa trên cơ sở hướng thành phần, cách thức thực hiện cụ thể từng bước, đặc biệt là đi sâu vào 2 bước chính, khó khăn nhất: mô hình hóa tính tin cậy của người phát triển thành phần và cách dự đoán tính tin cậy của kiến trúc sư phần mềm.

Ngoài ra, cũng hiểu được vì sao nên sử dụng mô hình Pham-Nordmann-Zhang vào dự đoán tính tin cậy. Đồng thời nắm được cách nâng cao tính tin cậy hệ thống sử dụng cấu trúc chịu lỗi FTSs.

Từ các kiến thức đã tìm hiểu ở trên, chúng ta đã bước đầu giải quyết được các vấn đề mà đồ án này đề ra, khi mà các hệ thống dự đoán tính tin cậy cho các hệ thống phần mềm hướng thành phần hiện tại vẫn chưa đáp ứng được:

* Bỏ qua lỗi lan truyền
* Bỏ qua mô hình chịu lỗi phần mềm
* Bỏ qua lỗi đồng thời

Tiếp theo, để hoàn thiện mục tiêu đã đặt ra ban đầu, trong chương tiếp theo, tôi sẽ trình bày về công cụ phát triển thực tế, áp dụng các kiến thức lý thuyết ở trên để có thể dự đoán được tính tin cậy cho các hệ thống phần mềm hướng thành phần.

# CHƯƠNG III. PHÁT TRIỂN CÔNG CỤ MÔ HÌNH HÓA ĐỘ TIN CẬY RMPI

## Giới thiệu công cụ RMPI

### Giới thiệu chung

Công cụ RMPI (Reliability Modeling, Prediction, and Improvements) là công cụ đã được phát triển dùng để dự đoán tính tin cậy cho các hệ thống phần mềm được TS.Phạm Thành Trung thực hiện, hiện đang được công khai toàn bộ Source code, tài liệu liên quan trên trang web: <http://rmpi.codeplex.com/>.

Một số đặc điểm chính của công cụ RMPI:

* Nhiệm vụ chính là dự đoán tính tin cậy cho các hệ thống phần mềm hướng thành phần đã được mô hình hóa.
* Sử dụng mô hình Markov để chuyển đổi mô hình tính tin cậy hệ thống do người phát triển phần mềm xây dựng.
* Hiện đang có giao diện dòng lệnh

### Cấu trúc dữ liệu XML đầu vào

Để sử dụng được công cụ RMPI, người phát triển thành phần phải xây dựng được mô hình tính tin cậy, sau đó kiến trúc sư phần mềm phải bổ sung thêm vào hồ sơ sử dụng của hệ thống để làm đầu vào cho công cụ.

Tài liệu này được viết trên ngôn ngữ xml. Lý do sử dụng ngôn ngữ xml:

* XML dễ hiểu. Dễ dàng tại các thẻ cho tài liệu, có thể tự do phát triển hệ thống phục vụ việc dự đoán tính tin cậy theo các tìm hiểu lý thuyết.
* XML có tổ chức. Giúp quá trình phân chia việc thiết kế hệ thống dễ dàng.
* XML là ngôn ngữ tiêu chuẩn quốc tế. Đồng nghĩa với việc mọi người trên thế giới đều có thể nhìn và hiểu được tài liệu viết bằng ngôn ngữ này.
* Công cụ RMPI có thể viết trên nhiều ngôn ngữ khác nhau. Trong trường hợp này tôi viết bằng Java, tuy nhiên nếu có những người muốn phát triển tiếp công cụ này trên các nền tảng công cụ khác vẫn hoàn toàn có thể thực hiện một cách dễ dàng việc đọc dữ liệu đầu vào.

Do những nguyên nhân nêu trên, ngôn ngữ xml được lựa chọn để viết tài liệu đầu vào cho công cụ RMPI.

Qua đây, tôi sẽ miêu tả sơ qua về cấu trúc dữ liệu XML này:

* Người phát triển thành phần hoàn thành mô hình tính tin cậy hệ thống:

<xs:element name="SystemReliabilityModeling">

<xs:complexType>

<xs:sequence>

<xs:element name="**ComponentReliabilitySpecification**">

<xs:complexType>

<xs:sequence>

<xs:element name="**FailureType**" maxOccurs="unbounded" minOccurs="0">

<xs:element name="**Service**" maxOccurs="unbounded" minOccurs="0">

<xs:element name="**Component**" maxOccurs="unbounded" minOccurs="0">

<xs:complexType>

<xs:sequence>

<xs:element name="**RequiredService**" maxOccurs="unbounded" minOccurs="0">

<xs:element name="**ProvidedService**" maxOccurs="unbounded" minOccurs="0">

<xs:element name="**ServiceImplementation**" maxOccurs="unbounded" minOccurs="0">

</xs:sequence>

<xs:attribute type="xs:string" name="name" use="optional"/>

</xs:complexType>

</xs:element>

</xs:sequence>

</xs:complexType>

</xs:element>

</xs:schema>

* Kiến trúc sư phần mềm bổ sung hồ sơ sử dụng của hệ thống:

<xs:element name="SystemReliabilityModel">

<xs:complexType>

<xs:sequence>

<xs:element name="**SystemArchitecture**">

<xs:complexType>

<xs:sequence>

<xs:element name="**ComponentInstance**" maxOccurs="unbounded" minOccurs="0">

<xs:element name="**ComponentConnector**" maxOccurs="unbounded" minOccurs="0">

<xs:element name="**UserInterface**">

</xs:element>

</xs:sequence>

</xs:complexType>

</xs:element>

<xs:element name="**UsageProfile**">

<xs:complexType>

<xs:sequence>

<xs:element name="**UsageProfilePart**">

</xs:element>

</xs:sequence>

</xs:complexType>

</xs:element>

</xs:sequence>

</xs:complexType>

</xs:element>

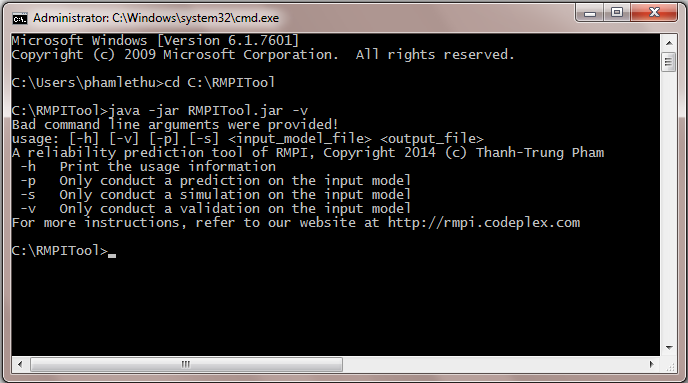
Trong công cụ tôi phát triển, tôi giữ nguyên cấu cấu file đầu vào xml của nêu trên.

Tuy nhiên so với phiên bản dòng lệnh, khi đưa một file xml đầu vào sai so với cấu trúc trên, chưa trực tiếp chỉ ra được lỗi mà phải đợi thực hiện các bước tiếp theo như khi bắt đầu dự đoán tính tin cậy mới có thể đưa ra thông báo file đầu vào bị lỗi. Chức năng này cũng đã được cung cấp và phát triển trong công cụ của tôi.

### Hình ảnh minh họa

Hướng dẫn cách chạy chương trình:

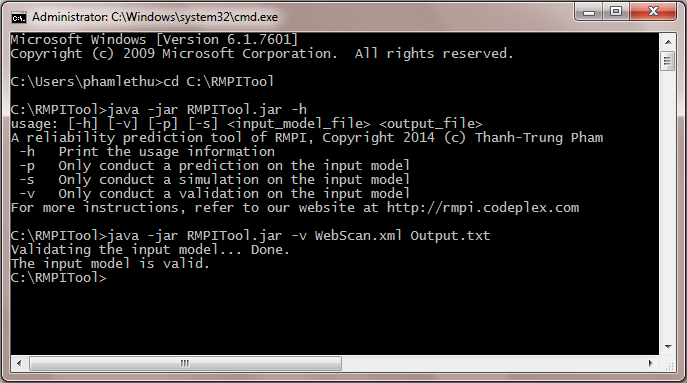
* Mở CommandPrompt, đổi đường dẫn đến thư mục chứa RMPITool và chạy "java -jar RMPITool.jar -h" để in hướng dẫn sử dụng.
* Với mỗi Case Study, sao chép file .xml tương ứng vào thư mục. Ví dụ (WebScan.xml)
* Chạy: "java -jar RMPITool.jar -v WebScan.xml Output.txt". File kết quả Output.txt sẽ thông báo file đầu vào .xml có hợp lệ hay không.
* Để lấy kết quả về tính tin cậy, chạy: "java -jar RMPITool.jar -p WebScan.xml Output.txt". Kết quả Output.txt chứa thông tin dự đoán tính tin cậy.
* Để mô phỏng kết quả (cho 1,000,000 lần thực hiện), chạy "java -jar RMPITool.jar -s WebScan.xml Output.txt". File Output.txt chứa kết quả mô phỏng. Chú ý thời gian thực hiện là khoảng 30 phút.



Hình 9. Màn hình chính giao diện dòng lệnh RMPI

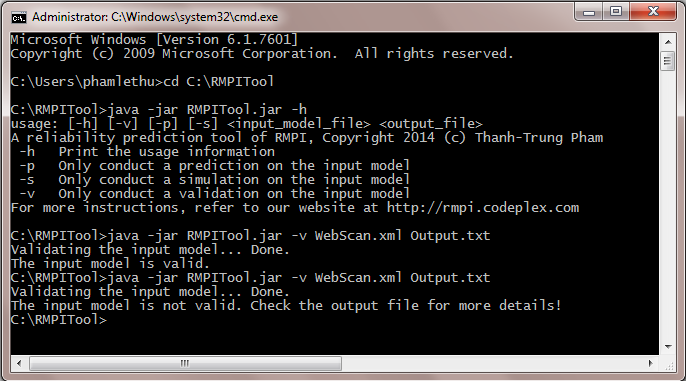
Để xác định chính xác của đánh giá đầu vào (WebScan.xml) với tùy chọn -v:

java -jar RMPITool.jar -v WebScan.xml Output.txt



Hình 10. Màn hình kiểm tra đầu vào giao diện dòng lệnh RMPI (chính xác)

Khi không đặt đầu vào WebScan.xml trong thư mục chứa file chạy.

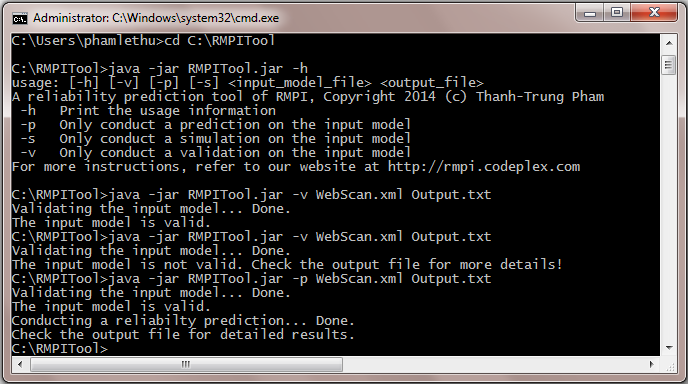


Hình 11. Màn hình kiểm tra đầu vào giao diện dòng lệnh RMPI (lỗi)

Chúng ta có thể thấy rằng đầu vào bị lỗi, và đầu ra Output.txt chứa “The input model is valid.” Cần phải sửa hoàn chỉnh lỗi này mới có thể tiến hành tiếp tục các bước tiếp theo.

Tiếp theo, để đánh giá tính tin cậy với tùy chọn -p:

java -jar RMPITool.jar -p WebScan.xml Output.txt



Hình 12. Màn hình dự đoán tính tin cậy giao diện dòng lệnh RMPI

Kết quả sẽ được tự động xuất ra một file txt trong thư mục chứa file chạy.

## Phát triển công cụ RMPI

### Ứng dụng mô hình PNZ

* Thành phần giữ lại từ công cụ RMPI
* Giữ nguyên hướng tiếp cận, tức là dự đoán tính tin cậy cho các hệ thống hướng thành phần 6 bước như trong chương II, mục 1 đã nêu.
* Giữ nguyên các loại cấu trúc điều khiển cơ sở như chương II, mục 3.1.1 đã nêu.
* Giữ nguyên các mô hình thất bại như chương II, mục 3.1.2
* Giữ nguyên cách chuyển đổi các thành phần hồ sơ sử dụng hệ thống

- Cấu trúc tuyến tính

- Cấu trúc rẽ nhánh

- Cấu trúc vòng lặp

- Cấu trúc song song

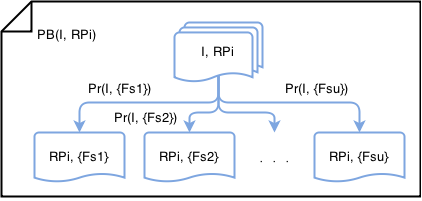
* Thành phần phát triển thêm
* Trong cách chuyển đổi thành phần hồ sơ sử dụng của hệ thống – cấu trúc thử lại

Xem xét một cấu trúc thử lại, lấy rc là biến đếm số lần thử lại, là tập các thất bại được xử lý, với tất cả IAIOS, FO(AFS AIOS) là mô hình thất bại cho IA tương đương của *thành phần thử lại* (viết tắt là RP).

Với mỗi đầu vào IAIOS của *cấu trúc thử lại*, quá trình chuyển đổi thay vì xây dựng một mô hình Markov bình thường, sẽ xây dựng một mô hình PNZ phản ánh tất cả các quá trình thực thi có thể của *cấu trúc thử lại* với đầu vào I và các xác suất tương ứng của nó, rồi sau đó xâu dựng mô hình thất bại cho IA tương đương từ mô hình PNZ. Với 3 bước như sau:

Bước 1: quá trình chuyển đổi sẽ xây dựng một khối PNZ cho mỗi lần thử lại. Khỗi PNZ của thử lại thứ I (PB(I,RPi)) phản ánh quá trình thực thi có thể của nó cho phát sinh thất bại. Nó mang theo nhãn “I,RPi” (gọi ngắn gọn thành [I,RPi]) là trạng thái đầu vào, trạng thái [RPi,F] cho tất cả các trạng thái phát sinh thất bại. Xác suất để đạt đến trạng thái [RPi,F] từ trạng thái [I,RPi] là cho tất cả các

Bước 2: quá trình chuyển đổi nhúng các khỗi PNZ này vào một mô hình PNZ đơn, mô hình đó phản ánh tất cả các thực thi có thể của *cấu trúc thử lại* với đầu vào như sau:



Hình 13. Khối PNZ

Khối PNZ cho mỗi thử lại thứ i

* Thêm một trạng thái [I,START]
* Thêm trạng thái [F] cho tất cả
* Thêm trạng thái [O] cho tất cả OAIOS
* Thêm chuyển tiếp từ trạng thái [I,START] đến trạng thái [I,RP0] với xác suất 1.0
* Với tất cả các khối PNZ PB(I,RPi) với , thêm chuyển tiếp từ trạng thái [I,RPi] tới trạng thái [O] với xác suất với mọi OAIOS. Đó là bởi vì một đầu ra chính xác (hoặc sai) của việc thực thi *thành phần thử lại* dẫn đến đầu ra chính xác (hoặc sai) của toàn bộ *cấu trúc thử lại.*
* Với khối PNZ PB(I,RPrc) (ví dụ: khối PNZ của thử lại cuối cùng), thêm chuyển tiếp từ trạng thái [RPrc,F] tới trạng thái [F] với xác suất 1.0 cho toàn bộ
* Với các khối PNZ khác, ví dụ: PB(I,RPi) với , thêm chuyển giao từ trạng tháo [RPi,F] tới trạng thái [I,RPi+1] với xác suất 1.0 nếu hoặc là tới trạng thái [F] với xác suất 1.0 với mọi .

Bước 3: sau khi việc chuyển đổi đã tạo được mô hình PNZ, mô hình thất bại cho IA tương đương được xây dựng như sau:

* Với mọi , là xác suất để đạt trạng thái tiếp thu [F] từ trạng thái tạm thời [I,START].
* Với mọi , là xác suất để đạt trạng thái tiếp thu [F] từ trạng thái tạm thời [I,START].

Ma trận chuyển đổi từ chuỗi các khối PNZ đã được tạo ra có dạng:

Trong đó Q là một ma trận vuông đại diện cho một bước chuyển giao giữa trạng thái hiện tại [I,START], [I,RPi] và [RPi,F] với mọi (với ); R đại diện cho một bước chuyển đổi từ trạng thái hiện tại để đạt được trạng thái [F] với mọi và [O] với mọi , I là một ma trận định danh với kích thước bằng số trạng thái tiếp thu.

Lấy B=(I-Q)-1R. Vì đâu là một chuỗi PNZ tiếp thu, thành phần bij của ma trận B là xác suất để chuỗi bị hấp thu vào trạng thái sj nếu nó bắt đầu từ trạng thái si. Do đó, mô hình thất bại của IA tương đương có thể chứa trong ma trận B.

* Trong cách chuyển đổi thành phần hồ sơ sử dụng của hệ thống – cấu trúc đa loại trừ

Xem xét một cấu trúc đa loại trừ, lấy n là số *thành phần loại trừ*, với mỗi , gọi là tập các thất bại đã được xử lý của *thành phần loại trừ* thứ i, với tất cả IAIOS, FO(AFS AIOS) là mô hình thất bại cho *thành phần loại trừ* thứ i (viết tắt là MPi).

Cũng giống như *cấu trúc thử lại,* với mỗi đầu vào IAIOS của *cấu trúc đa loại trừ*, quá trình chuyển đổi thay vì xây dựng một mô hình Markov bình thường, sẽ xây dựng một một mô hình PNZ phản ánh tất cả các quá trình thực thi có thể của *cấu trúc đa loại trừ* với đầu vào I và các xác suất tương ứng, sau đó xây dựng mô hình thất bại cho IA tương đương từ mô hình PNZ. Với 3 bước như sau:

Bước 1: quá trình chuyển đổi xây dựng một khối PNZ từ mỗi *thành phần đa loại trừ.* Khối PNZ cho *thành phần đa lại trừ* thứ i (MB(I,MPi)) phản ánh quá trình thực thi có thể của nó để tạo thất bại. Nó chứa trạng thái [I,MPi] làm trạng thái đàu vào, trạng thái [MPi,F] với mị là trạng thái tạo thất bại. Xác suất để đạt đến trạng thái [MPi,F] từ trạng thái [I,MPi] là với mọi .

Bước 2: quá trình chuyển đổi nhúng các khỗi PNZ vào một mô hình PNZ duy nhất. Mô hình đó phản ánh tất cả các quá trình thực thi có thể của cấu trúc đa loại trừ với đầu vào như sau:

* Thêm một trạng thái [I,START]
* Thêm trạng thái [F] với mọi .
* Thêm trạng thái [O] với mọi .
* Thêm một chuyển đổi từ trạng thái [I,START] đến trạng thái [I,MPi] với xác suất 1.0
* Với mọi khối PNZ PB(I,MPi) trong đó , thêm chuyển đổi từ trạng thái [I,MPi] tới trạng thái [O] với xác suất với mọi . Nó đưa ra kết quả đúng (hoặc sai) cho mỗi thực thi *thành phần đa loại trừ* dẫn đến kết quả đúng (hoặc sai) của toàn bộ *cấu trúc đa loại trừ.*
* Với khối PNZ PB(I,MPn) (ví dụ: khối PNZ của *thành phần đa loại trừ* cuối cùng), thêm một chuyển giao từ trạng thái [MPn,F] đến trạng thái [F] với xác suất 1.0 cho mọi .
* Với các khối PNZ khác, ví dụ PB(I,MPi) với , thêm chuyển giao từ trạng thái [MPn, F] tới hoặc là trạng thái [I,MPx] với xác suất 1.0 trong đó là chỉ số thấp nhất thỏa mãn hoặc là trạng thái [F] với xác suất 1.0 nếu ko có chỉ số nào thoải mãi với mọi .

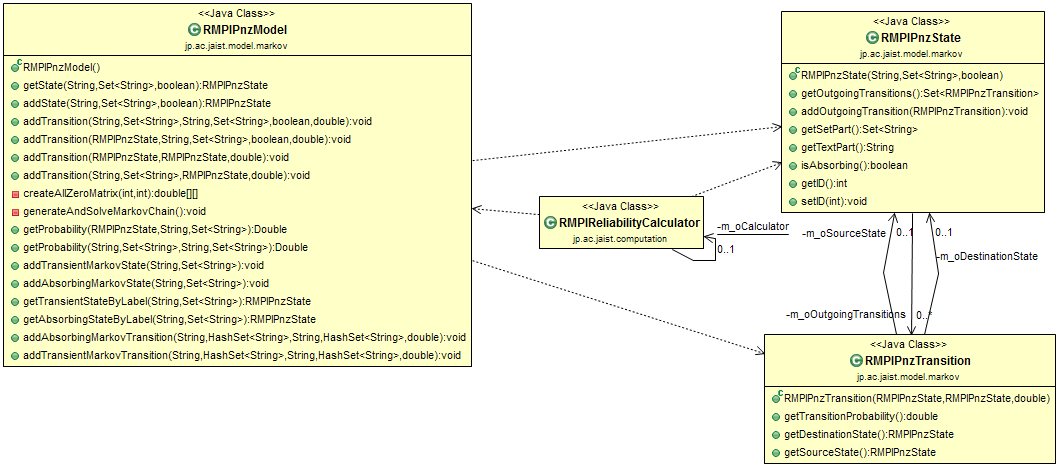
Bước 3: sau khi việc chuyển đổi đã tạo được mô hình PNZ, mô hình thất bại cho IA tương đương được xây dựng như sau:

* Với mọi , là xác suất để đạt trạng thái tiếp thu [F] từ trạng thái tạm thời [I,START].
* Với mọi , là xác suất để đạt trạng thái tiếp thu [F] từ trạng thái tạm thời [I,START].

Cuối cùng, trong hướng tiếp cận mà tôi tìm hiểu, tôi định nghĩa tính tin cậy là

R=1-POFOD

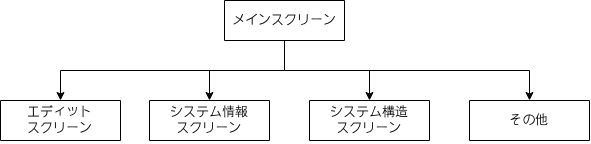
Trong đó POFOD là xác suất của thất bại cần đạt được, được đưa ra bởi đầu vào của dịch vụ được cung cấp bởi hệ thống chạy chính xác. Do đó, tính tin cậy cho dịch vụ được cung cấp mà cách sử dụng hệ thống hiện tại đang trỏ đến là xác suất mà dịch vụ đó cung cấp một đầu ra chính xác từ đầu vào chính xác, nó chính là của mô hình thất bại cho IA tương đương của quá trình thực thi dịch vụ này.



Hình 14. Sơ đồ UML cho thành phần bổ sung của RMPI

### Phát triển giao diện đồ họa người dùng

Bên cạnh việc thay đổi hướng tiếp cận cho các dự đoán tính tin cậy của các hệ thống phần mềm hướng thành phần từ sử dụng mô hình Markov sang mô hình PNZ, để công cụ RMPI có thể hoạt động hiệu quả và thân thiện với người dùng hơn, trong phạm vi đồ án này tôi đã phát triển thêm giao diện đồ họa cho công cụ.

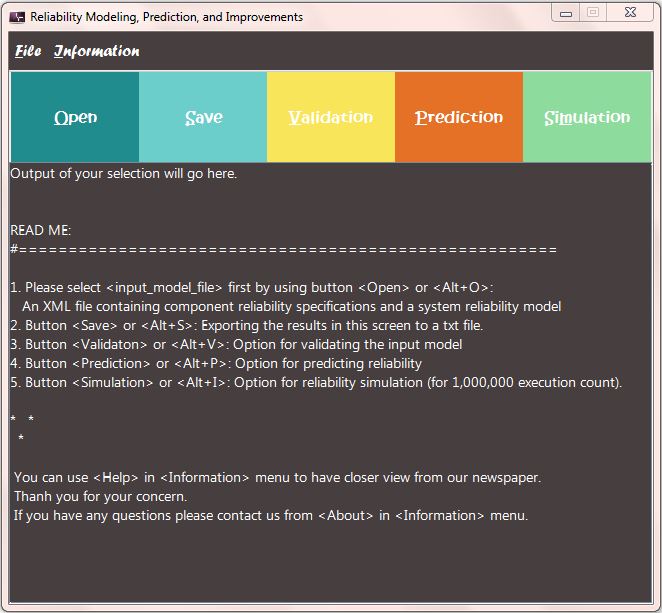


Hình 15. Sơ đồ cấu trúc giao diện đồ họa RMPI

* Với màn hình Edit: hỗ trợ người sử dụng có thể trực tiếp sửa vào file đầu vào xml từ đó nhìn thấy sự thay đổi tính tin cậy hệ thống khi thay đổi xác suất gặp lỗi của các thành phần.
* Với màn hình System’s Information: giúp người sử dụng nhìn được các thành phần cơ bản của hệ thống như: UI, tên các dịch vụ, các loại lỗi.
* Với màn hình System’s Structure: vẽ ra cấu trúc hệ thống một cách đơn giản, dễ nhìn nhất để người sử dụng ngay lập tức hiểu được cấu trúc của toàn bộ hệ thống file đầu vào.
* Các màn hình khác bao gồm: Màn hình giới thiệu chung về công cụ, màn hình tài liệu hướng dẫn cách sử dụng công cụ cũng như giải thích cách hoạt động của công cụ.

### Hình ảnh minh họa

Một số hình ảnh minh họa cho công cụ tôi phát triển.



**1**

**6**

**5**

**4**

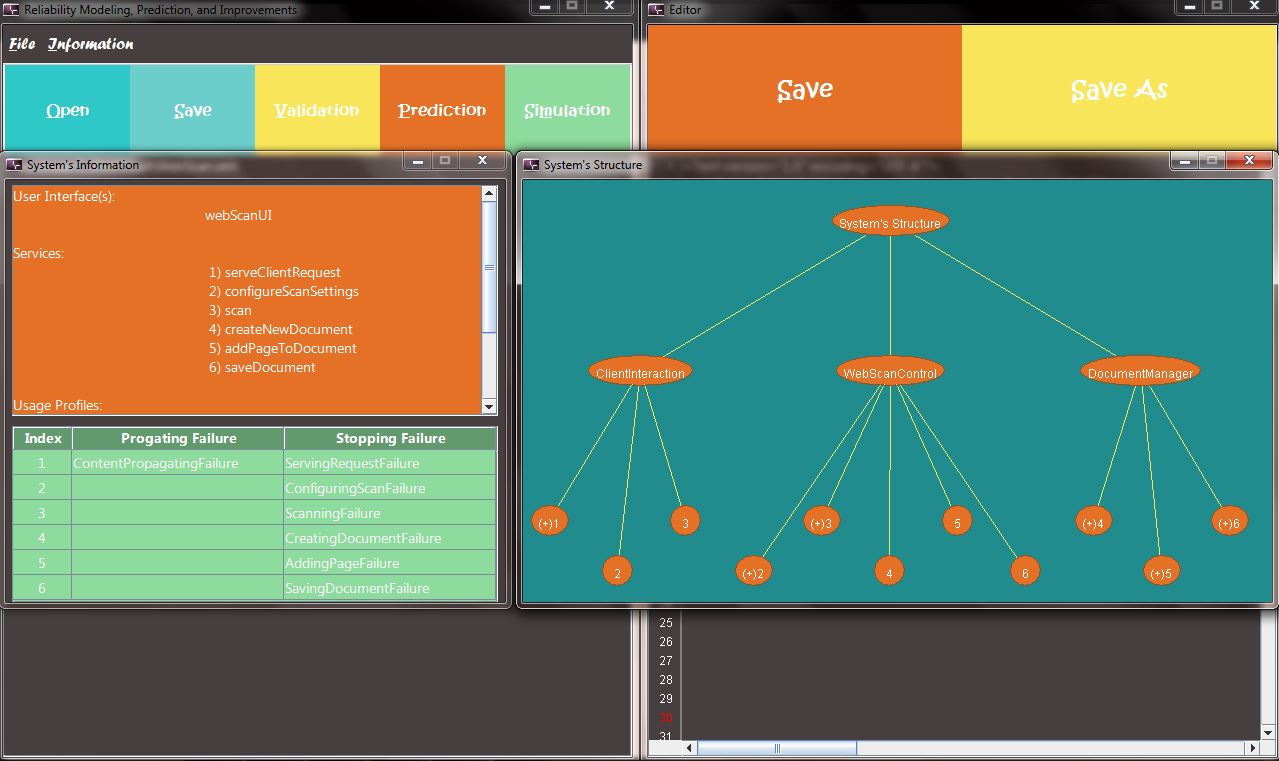
**3**

**2**

**7**

Hình 16. Màn hình chính giao diện đồ họa RMPI

1. Menu Bar: chứa các lệnh Exit, Help, About.
2. Nút Open: Mở cửa sổ để người sử dụng chọn file xml đầu vào. (Có bộ lọc các file xml)
3. Nút Save: Save toàn bộ nội dung trong TextArea bên dưới ra file txt.
4. Nút Validation: Kiểm tra xem file xml đầu vào có tương ứng với file XSD (XML schema definition) cho trước hay không.
5. Nút Prediction: Dự đoán tính tin cậy hệ thống
6. Nút Simulation: Mô phỏng việc chạy hệ thống thực tế để so sánh với kết quả dự đoán tính tin cậy đã có.
7. Giới thiệu khái quát cách sử dụng công cụ RMPI.



Hình 17. Màn hình sau khi mở file xml giao diện đồ họa RMPI

## Thử nghiệm và đánh giá công cụ

Hệ thống WebScan: hệ thống cho phép người sử dụng tại máy tính để bàn scan một hoặc nhiều hình ảnh vào hệ thống quản lý tài liệu sử dụng trình duyệt, ví dụ như Internet Explorer

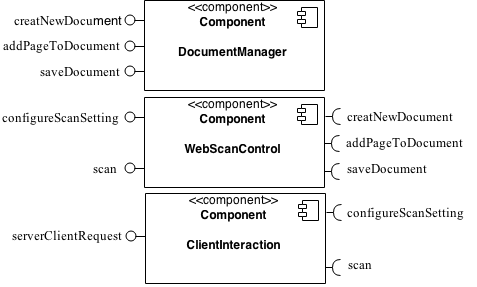
### Mô tả bài toán thử nghiệm

#### Các dịch vụ, các thành phần, các thực thi của hệ thống

Đầu tiên, chúng ta chia hệ thống thành 6 dịch vụ:

* serveClientRequest
* configureScanSettings
* scan
* createNewDocument
* addPageToDocument
* saveDocument

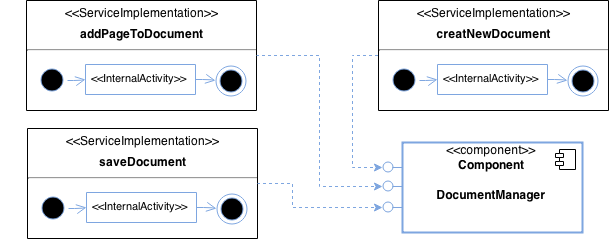
Tiếp theo, chia các dịch vụ thành các thành phần thích hợp:



Hình 18. Các thành phần của hệ thống WebScan

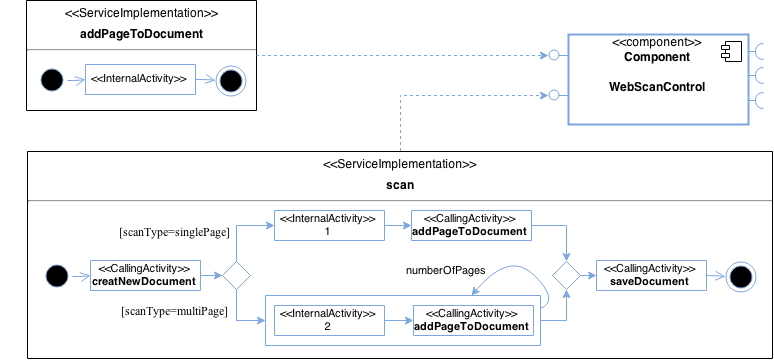
Tiếp theo, xây dựng thực thi dịch vụ cho từng thành phần đã chia ở trên

- Cho module DocumentManager



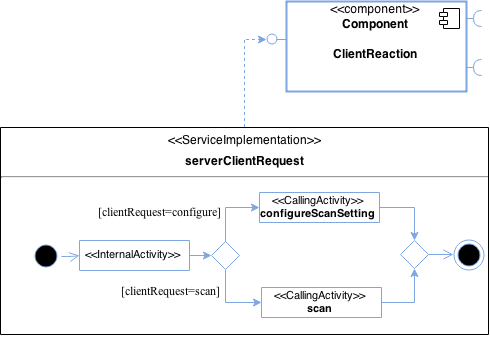
Hình 19. Các thực thi dịch vụ của thành phần DocumentManager

- Cho module WebScanControl



Hình 20. Các thực thi dịch vụ của thành phần WebScanControl

- Cho module ClientInteraction



Hình 21. Các thực thi dịch vụ của thành phần ClientReaction

#### Các mô hình thất bại của hệ thống

Đầu tiên, mô hình hóa tất cả các loại lỗi bao gồm cả lỗi lan truyền và lỗi dừng

- Đối với lỗi lan truyền

|  |  |
| --- | --- |
| **Loại lỗi lan truyền** | **Ký hiệu** |
| ContentPropagatingFailure | FP1 |

Bảng 2. Các loại lỗi lan truyền của hệ thống WebScan

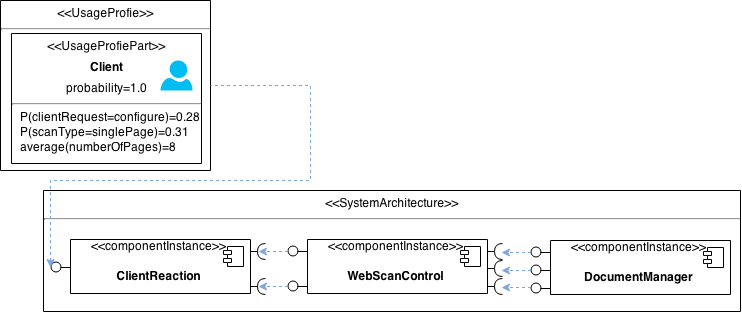
- Đối với lỗi dừng

|  |  |
| --- | --- |
| **Loại lỗi dừng** | **Ký hiệu** |
| ServingRequestFailure | FS1 |
| ConfiguringScanFailure | FS2 |
| ScanningFailure | FS3 |
| CreatingDocumentFailure | FS4 |
| AddingPageFailure | FS5 |
| SavingDocumentFailure | FS6 |

Bảng 3. Các loại lỗi dừng của hệ thống WebScan

#### Kiến trúc tổng quát của hệ thống

Đầu tiên, ta xác định cấu trúc tổng quát của toàn bộ hệ thống:



Hình 22. Cấu trúc hệ thống WebScan

Tiếp theo, ta định nghĩa các Instance tương ứng cho mỗi thành phần:

|  |  |
| --- | --- |
| **Thành phần** | **Instance** |
| ClientInteraction | clientInteraction |
| WebScanControl | webScanControl |
| DocumentManager | documentManager |

Bảng 4. Các Instances cho các thành phần của hệ thống WebScan

Tiếp tục định nghĩa các kêt nỗi giữa các module (Theo ngôn ngữ xml), ta có:

<tns:ComponentConnector>

<tns:FromService fromProvidedService="createNewDocument" fromComponentInstance="documentManager"/>

<tns:ForService forRequiredService="createNewDocument" forComponentInstance="webScanControl"/>

</tns:ComponentConnector>

<tns:ComponentConnector>

<tns:FromService fromProvidedService="addPageToDocument" fromComponentInstance="documentManager"/>

<tns:ForService forRequiredService="addPageToDocument" forComponentInstance="webScanControl"/>

</tns:ComponentConnector>

<tns:ComponentConnector>

<tns:FromService fromProvidedService="saveDocument" fromComponentInstance="documentManager"/>

<tns:ForService forRequiredService="saveDocument" forComponentInstance="webScanControl"/>

</tns:ComponentConnector>

<tns:ComponentConnector>

<tns:FromService fromProvidedService="scan" fromComponentInstance="webScanControl"/>

<tns:ForService forRequiredService="scan" forComponentInstance="clientInteraction"/>

</tns:ComponentConnector>

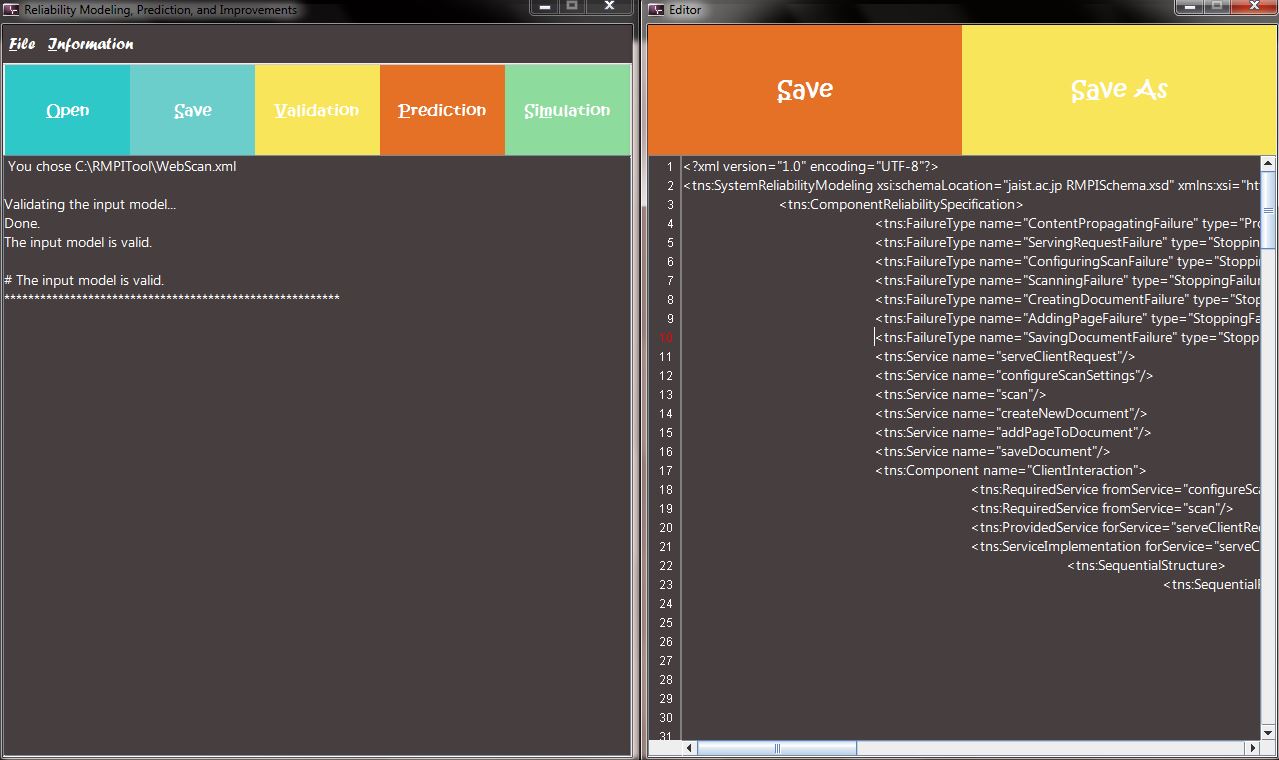
<tns:ComponentConnector>

<tns:FromService fromProvidedService="configureScanSettings" fromComponentInstance="webScanControl"/>

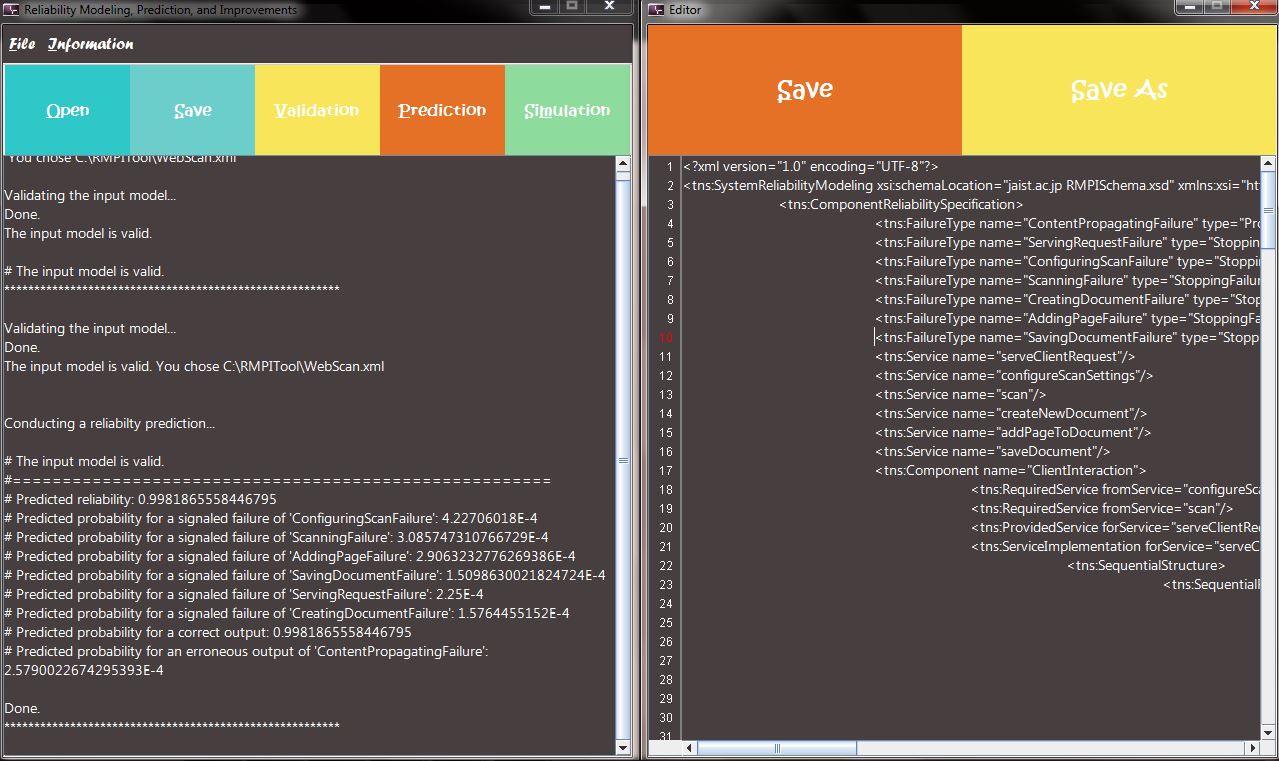
<tns:ForService forRequiredService="configureScanSettings" forComponentInstance="clientInteraction"/>

</tns:ComponentConnector>

### Tính toán sử dụng công cụ RMPI



Hình 23. Màn hình kiểm tra đầu vào giao diện đồ họa RMPI



Hình 24. Màn hình dự đoán tính tin cậy giao diện đồ họa RMPI

Kết quả dự đoán tính tin cậy hệ thống WebScan, ta có:

# The input model is valid.

#====================================================================

# Predicted reliability: 0.9981865558446795

# Predicted probability for a signaled failure of 'CreatingDocumentFailure': 1.5764455152E-4

# Predicted probability for a signaled failure of 'ConfiguringScanFailure': 4.22706018E-4

# Predicted probability for a signaled failure of 'ServingRequestFailure': 2.25E-4

# Predicted probability for a signaled failure of 'AddingPageFailure': 2.9063232776269386E-4

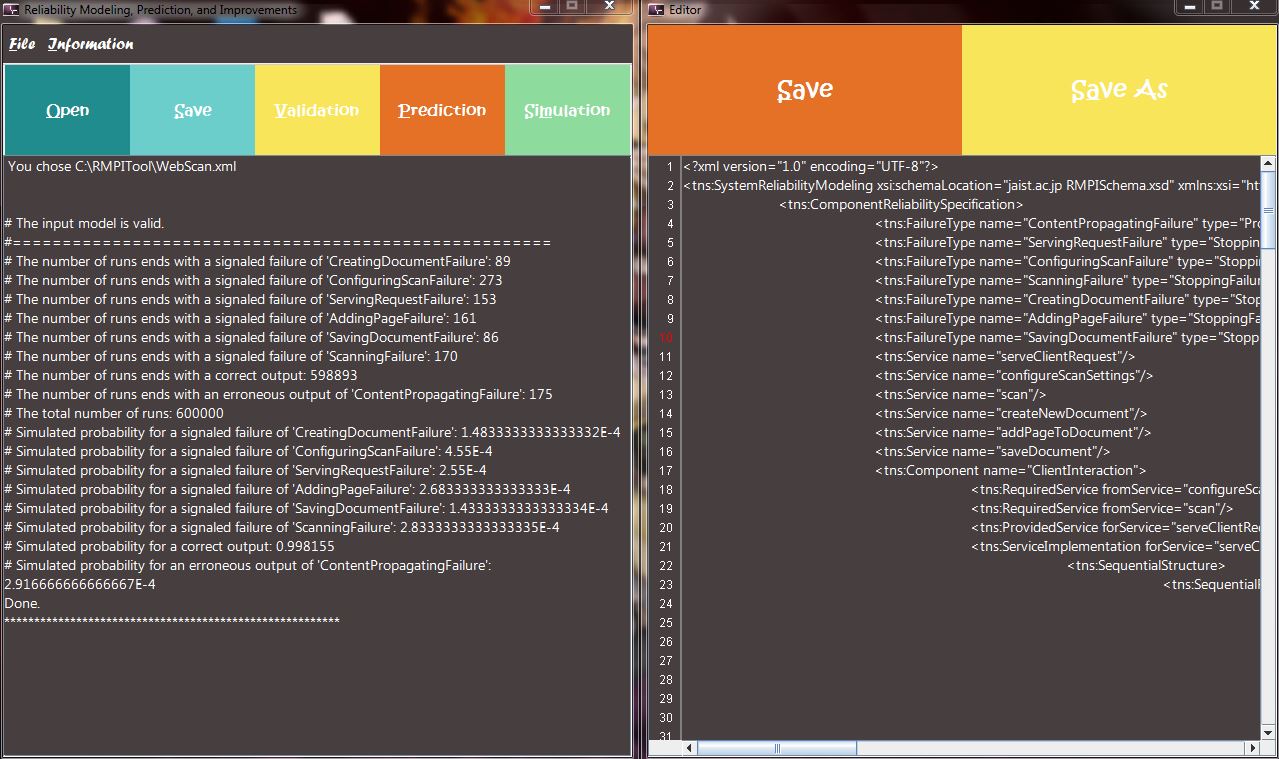
# Predicted probability for a signaled failure of 'SavingDocumentFailure': 1.5098630021824724E-4

# Predicted probability for a signaled failure of 'ScanningFailure': 3.085747310766729E-4

# Predicted probability for a correct output: 0.9981865558446795

# Predicted probability for an erroneous output of 'ContentPropagatingFailure': 2.5790022674295393E-4

Để kiếm chứng kết quả trên đã đảm bảo chính xác hay chưa, công cụ RMPI đã được cung cấp chức năng “Simulation”, cách thực hiện chức năng này đã được trình bày cụ thể trong Chương 2, mục 4.3:



Hình 25. Màn hình mô phỏng tính tin cậy (simulation) giao diện đồ họa RMPI

Sự sai số giữa dự đoán tính tin cậy và mô phòng tính tin cậy được trình bày ở bảng 5 dưới đây:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Số lần mô phỏng** | **Dự đoán tính tin cậy** | **Mô phỏng tính tin cậy** | **Sai số** | **Sai số (%)** |
| 200000 | 0.998187 | 0.9983 | 0.000113 | 0.013 |
| 400000 | 0.9982375 | 0.0000505 | 0.00505 |
| 600000 | 0.998155 | 0.000032 | 0.0032 |

Bảng 5. Sai số giữa dự đoán tính tin cậy và mô phỏng tính tin cậy

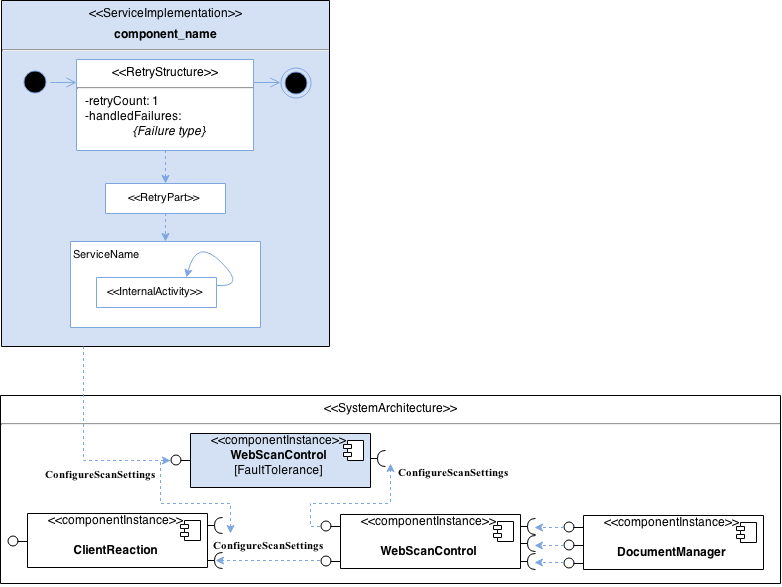
Ta có sơ đồ như sau:

Hình 26. Đồ thị tỷ lệ xuất hiện các loại lỗi(%) của hệ thống WebScan

🡺 Lỗi FS2: “ConfiguringScanFailure” có tỷ lệ xuất hiện nhiều nhất.

Làm theo 3 bước được nêu ra trong phần lý thuyết Chương II, mục 5 bên trên:

Bước 1: Mô hình module chịu lỗi



Hình 27. Mô hình module chịu lỗi

Bước 2: Tạo instance cho module

|  |  |
| --- | --- |
| **Module** | **Instance** |
| WebScanControlFaultTolerance | webScanControlFaultTolerance |

Bảng 6. Instance cho thành phần chịu lỗi của hệ thống WebScan

Bước 3: Sửa lại kết nối đến các module

Theo ngôn ngữ xml, ta có:

<tns:ComponentConnector>

<tns:FromService fromProvidedService="configureScanSettings" fromComponentInstance="webScanControl"/>

<tns:ForService forRequiredService="configureScanSettings" forComponentInstance="webScanControlFaultTolerance"/>

</tns:ComponentConnector>

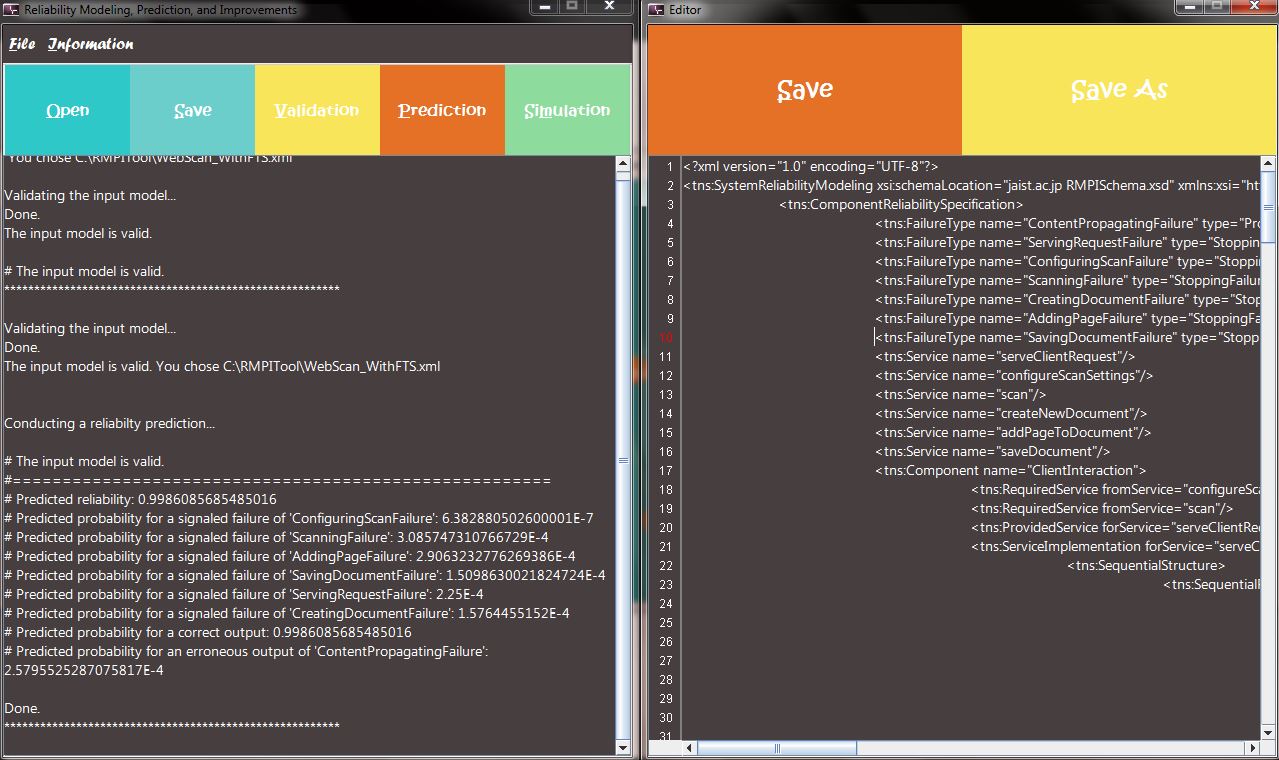
<tns:ComponentConnector>

<tns:FromService fromProvidedService="configureScanSettings" fromComponentInstance="webScanControlFaultTolerance"/>

<tns:ForService forRequiredService="configureScanSettings" forComponentInstance="clientInteraction"/>

</tns:ComponentConnector>

Sau khi đã hoàn thành việc thay đổi cấu trúc hệ thống như trên, chúng ta sử dụng tool RMPI để so sánh kết quả:



Hình 28. Màn hình dự đoán tính tin cậy sau khi sử dụng cấu trúc chịu lỗi

Khi đó, ta có kết quả:

# The input model is valid.

#====================================================================

# Predicted reliability: 0.9986085685485016

# Predicted probability for a signaled failure of 'CreatingDocumentFailure': 1.5764455152E-4

# Predicted probability for a signaled failure of 'ConfiguringScanFailure': 6.382880502600001E-7

# Predicted probability for a signaled failure of 'ServingRequestFailure': 2.25E-4

# Predicted probability for a signaled failure of 'AddingPageFailure': 2.9063232776269386E-4

# Predicted probability for a signaled failure of 'SavingDocumentFailure': 1.5098630021824724E-4

# Predicted probability for a signaled failure of 'ScanningFailure': 3.085747310766729E-4

# Predicted probability for a correct output: 0.9986085685485016

# Predicted probability for an erroneous output of 'ContentPropagatingFailure': 2.5795525287075817E-4

So sánh kết quả tính tin cậy cũng như tỷ lệ xuất hiện các lỗi của hệ thống WebScan trước và sau khi thêm thành phần chịu lỗi FTS, ta có bảng như sau:

|  |  |  |
| --- | --- | --- |
|  | **WebScan** | **WebScan\_WithFTS** |
| Reliability | 0.9981865558446795 | 0.9986085685485016 |
| CreatingDocumentFailure | 1.5764455152E-4 | 1.5764455152E-4 |
| ConfiguringScanFailure | 4.22706018E-4 | 6.382880502600001E-7 |
| ServingRequestFailure | 2.25E-4 | 2.25E-4 |
| AddingPageFailure | 2.9063232776269386E-4 | 2.9063232776269386E-4 |
| SavingDocumentFailure | 1.5098630021824724E-4 | 1.5098630021824724E-4 |
| ScanningFailure | 3.085747310766729E-4 | 3.085747310766729E-4 |
| ContentPropagatingFailure | 2.5790022674295393E-4 | 2.5795525287075817E-4 |

Bảng 7. Bảng so sánh kết trả trước và sau khi sử dụng cấu trúc chịu lỗi FTSs

### Nhận xét, đánh giá công cụ RMPI

Công cụ RMPI đã hoàn thành được các nhiệm vụ đặt ra ban đầu của đồ án:

* + - Công cụ đã kiểm tra được dữ liệu đầu vào XML có phù hợp với XSD tương ứng được quy định trước hay không. Trong trường hợp sai đã đưa ra được thông báo và hướng dẫn cho người sử dụng.
    - Công cụ đã dự đoán được tính tin cậy hệ thống: Đưa ra được con số cụ thể về tính tin cậy.
    - Công cụ đã đưa ra được xác suất xuất hiện các loại lỗi của hệ thống.
    - Công cụ đã cho phép trực tiếp sửa vào file .xml đầu vào để nhìn thấy được sự thay đổi của tính tin cậy. (Trong trường hợp áp dụng công nghệ cấu trúc chịu lỗi đúng, đảm bảo được tính tin cậy của hệ thống tăng lên). Với thử nghiệm là hệ thống WebScan trong mục 3 này, ta thấy:

- Dự đoán tính tin cậy của hệ thống tăng khoảng 0.042277%

- Tỷ lệ xuất hiện lỗi “ConfiguringScanFailure” giảm khoảng 99.84899954%

* + - Khi mở file .xml đầu vào, hiển thị được toàn bộ thông tin các thành phần, các loại lỗi, cấu trúc của toàn bộ hệ thống để người sử dụng dễ hình dung.
    - Sử dụng chức năng “Simulation” của công cụ RMPI chúng ta có thể xác định được tính chính xác về lý thuyết và công cụ RMPI đã thực hiện. Trong trường hợp của hệ thống WebScan mức độ chênh lệch, sai số giữa dự đoán tính tin cậy và mô phỏng tính tin cậy đã được thể hiện trong bảng 5:

- Số vòng chạy càng tăng, giá trị mô phỏng tính tin cậy càng gần giá trị dự đoán tính tin cậy.

- Sai số lớn nhất là 0.013%

🡪 Có thể chấp nhận về tính chính xác theo lý thuyết của công cụ RMPI.

Tuy nhiên, công cụ RMPI vẫn còn hạn chế như:

* Hiện tại người phát triển vẫn phải tự viết file đầu vào cho công cụ RMPI. (Do đó, người phát triển thành phần buộc phải hiểu rõ cách hoạt động của công cụ RMPI cũng như cấu trúc của XSD đã được quy định.)

## Kết chương

Trong chương này chúng ta áp dụng toàn bộ nội dung lý thuyết đã được nêu ra ở chương II để phát triển công cụ RMPI.

Đầu tiên, chúng ta nắm được công cụ RMPI nền tảng, những thiết sót và điểm cần khắc phục. Sau đó, tôi trình bày các cách phát triển công cụ đi theo yêu cầu và nhiêm vụ của đồ án đã được đặt ra từ đâu.

Đặc biệt, với thử nghiệm là hệ thống WebScan, chúng ta nắm rõ cách thức dự đoán tính tin cậy cho hệ thống hướng thành phần bao gồm 6 bước. Đồng thời cũng hiểu được mô hình PNZ và cách các áp dụng mô hình PNZ vào dự đoán tính tin cậy.

Để sử dụng được công cụ này, hướng tìm hiểu của tôi yêu cầu:

* Người phát triển thành phần viết tài liệu đặc tả tính tin cậy.
* Kiến trúc sư phần mềm bổ sung vào tài liệu đó hồ sơ sử dụng của hệ thống

🡪 Từ đó, ta có được đầu vào là file xml cho công cụ RMPI. (Phải đáp ứng đúng các yêu cầu như miêu tả trong file XSD đã được định nghĩa.)

Sau đó, các thành phần của hệ thống đã được mô tả trong file đầu vào này sẽ được chuyển đổi một cách tự động sang mô hình PNZ để dự đoán tính tin cậy. (Cách thức chuyển đổi, và tính toán tính tin cậy đã được để cập ở chương II)

Cuối cùng, để nâng cao tính tin cậy vừa dự đoán được, (trong trường hợp của hệ thống WebScan), người phát triển thành phần sẽ bổ sung thêm một thành phần có chứa cấu trúc chịu lỗi.

Nhờ vào việc áp dụng lý thuyết vào công cụ thực tế như vậy, chúng ta có thể tạo được các hệ thống phần mềm đảm bảo tính tin cậy theo con đường tiết kiệm chi phí nhất do chúng ta có thể tránh được việc phải sử dụng một khoản tiền lớn cho công việc thay đổi phần mềm ở những giai đoạn cuối về sau này.

# KẾT LUẬN

## Kết quả đạt được

* Ưu điểm

Hoàn thành được mục tiêu và nhiệm vụ đặt ra cho đồ án:

* Hiểu được cách dự đoán tính tin cậy dựa trên cơ sở hướng thành phần.
* Hiểu được mô hình Pham-Nordmann-Zhang.
* Hoàn thành việc áp dụng mô hình Pham-Nordmann-Zhang vào dự đoán tính tin cậy cho các hệ thống phần mềm hướng thành phần.
* Hiểu được cách nâng cao tính tin cậy bằng sử dụng công nghệ cấu trúc chịu lỗi.
* Phát triển thành công công cụ thực tế (Công cụ RMPI - Reliability Modeling, Prediction, and Improvements).
* Nhược điểm

Do thời gian, dữ liệu và nhân lực còn có hạn chế nên đồ án hiện tại vãn còn những nhược điểm như sau:

* Công cụ RMPI vẫn còn điểm khó sử dụng, cụ thể đã được phân tích và đánh giá ở Chương 3, mục 3.3.
* Các chức năng còn đơn giản.
* Hiện tại trong đồ án này, tôi chưa nêu ra được nếu so sánh với các phương pháp khác, dự đoán tính tin cậy cho các hệ thống phần mềm hướng thành phần đã thực sự chính xác và hiệu quả hay chưa.
* Tương tự, do thiếu sót về dữ liệu nên chưa thể so sánh với các phương pháp khác để xem mức độ cải thiện độ tin cậy cho khi sử dụng cấu trúc chịu lỗi FTSs đã thực sự hiệu quả chưa.

## Hướng phát triển

Để phát triển hướng nghiên cứu này, tôi dự định sẽ:

* Phát triển hướng tiếp cận ra cho các lỗi lan truyền phức tạp hơn.
* Phát triển thêm nhiều cấu trúc chịu lỗi FTSs hơn.
* Sau khi dự đoán được tính tin cậy cho hệ thống phần mềm hướng thành phần, ngay lập tức vẽ ra đồ thị để xem tỉ lệ xuất hiện các loại lỗi trong phần mềm, từ đó dễ dàng nhìn thấy thành phần nào đang có tỉ lệ xuất hiện lỗi nhiều nhất.
* Tối ưu giao diện hơn nữa.
* Ngoài ta tôi còn dự định tiếp tục phát triển sơ đồ mô hình hóa tính tin cậy và công cụ dự đoán tính tin cậy để có thể giúp người phát triển phần mềm tự động viết được tài liệu đặc tả tính tin cậy.

Bằng việc đi theo hướng phát triển đã nêu, hướng nghiên cứu cũng như công cụ mà tôi đã phát triển có thể tăng khả năng áp dụng vào thực tế.

# TÀI LIỆU THAM KHẢO

[1] ISO/IEC-25010:2011, “Systems and software quality requirements and evaluation (square) - system and software quality models (square)”, 2011.

[2] R. Rana, “Defect prediction & prevention in automotive software development”, 2013.

[3] R. Roshandel, *Calculating architectural reliability via modeling and analysis*. PhD thesis, University of Southern California, 2006.

[4] X. CHENGJIE, *Availability and Reliability Analysis of Computer Software Systems Considering Maintenance and Security Issues*. PhD thesis, 2011.

[5] F. Brosch, *Integrated software architecture-based reliability prediction for IT systems*, vol. 9. KIT Scienti c Publishing, 2012.

[6] M. Larsson, *Predicting quality attributes in component-based software systems*. Mälardalen University, 2004.

[7] T.-T. Pham and X. Defago, “Reliability prediction for component-based software systems with architectural-level fault tolerance mechanisms”, in *Availability, Reliability and Security* (ARES), *2013 Eighth International Conference on* , pp. 11-20, IEEE, 2013.

[8] H. Pham, *System software reliability*. Springer, 2006.

[9] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, “Basic concepts and taxonomy of dependable and secure computing”, *Dependable and Se- cure Computing, IEEE Transactions on*, vol. 1, no. 1, pp. 11-33, 2004.

[10] L. L. Pullum, *Software fault tolerance techniques and implementation*. Artech House, 2001.

[11] Cheung, R.C., A user-oriented software reliability model, *IEEE Transactions on Software Engineering*, pp. 118-125, 1980.

[12] V. Cortellessa, H. Singh, B. Cukic, Early reliability assessment of UML based software models, in: Proceedings of the 3rd international workshop on Software and performance, ACM, Rome, Italy, pp. 302-309, 2002.

[13] A. Filieri, C. Ghezzi, V. Grassi, R. Mirandola, Reliability analysis of component-based systems with multiple failure modes, in: Proceedings of the 13th international conference on Component-Based Software Engineering, CBSE’10, pp. 1-20, 2010.

[14] A. Mohamed, M. Zulkernine, On failure propagation in component-based software systems, in: Proceedings of the 2008 The Eighth International Conference on Quality Software, IEEE Computer Society, pp. 402-411, 2008.

[15] H.Pham, L.Nordmann, and Z. Zhang, “A general imperfect-softwaredebugging model with s-shaped fault-detection rate”, *Reliability, IEEE Transactions on*, vol. 48, no. 2, pp. 169-175, 1999.

[16] J. Voas, “Maintaining component-based systems”, *Software, IEEE*, vol. 15, no. 4, pp. 22-27, 1998.

[17] R. H. Reussner, H. W. Schmidt, and I. H. Poernomo, “Reliability prediction for component-based software architectures”, Journal of systems and software, vol. 66, no. 3, pp. 241-252, 2003.

[18] A. Avizienis, “Fault-tolerance and fault-intolerance: Complementary approaches to reliable computing”, in ACM SIGPLAN Notices, vol. 10, pp. 458-464, ACM, 1975.

[19] W. R. Elmendorf, “Fault-tolerant programming”, in *Proceedings of the 2nd International Symposium on Fault-tolerant Computing (FTCS-2)*, vol. 31, pp. 79-83, 1972.

[20] H. Kopetz, “Software redundancy in real time systems.”, in *IFIP Congress*, vol. 74, pp. 182-186, 1974.

[21] E. Girard and J. Rault, “A programming technique for software reliability”, in *Proceedings of the 1973 IEEE Symposium on Computer Software Reliability*, pp. 44-50, 1973.

[22] V. S. Sharma, K. S. Trivedi, Reliability and performance of component based software systems with restarts, retries, reboots and repairs, in: Proceedings of the 17th International Symposium on Software Reliability Engineering, IEEE Computer Society, pp. 299-310, 2006.

[23] W.-L. Wang, D. Pan, M.-H. Chen, Architecture-based software reliability modeling, J. Syst. Softw. 79 (1) 132-146, 2006.