ĐỀ XUẤT ĐỀ TÀI KHOA HỌC VÀ CÔNG NGHỆ CẤP BỘ  
NĂM 2017

# Tê đề tài:

**PHÁT HIỆN VÀ CẢI THIỆN MÃ NGUỒN XỬ LÝ NGOẠI LỆ XẤU  
TRONG JAVASCRIPT**

# Lĩnh vực nghiên cứu:

Tự nhiên: 🞏 Kỹ thuật: 🗹 Môi trường: 🞏

Kinh tế, XHNV: 🞏 Nông lâm: 🞏 ATLĐ: 🞏

Giáo dục: 🞏 Y Dược: 🞏 Sở hữu trí tuệ: 🞏

# Tính cấp thiết:

Hiện nay, các sản phẩm của công nghệ phần mềm đã và đang được ứng dụng vào việc quản lý, điều hành và tác nghiệp trong hầu hết các lĩnh vực của đời sống và đem lại hiệu quả kinh tế cao. Do sự tiện lợi của internet trong xu thế toàn cầu, tốc độ đường truyền đã được cải thiện cũng như sự phát triển của công nghệ nên những sản phẩm ứng dụng được xây dựng trên nền tảng World Wide Web ngày càng phổ biến. Cùng với sự phát triển của ngôn ngữ đánh dấu siêu văn bản thế hệ mới - HTML5, JavaScript đã tạo ra sức mạnh cho các ứng dụng chạy trên nền Web. JavaScript là một ngôn ngữ lập trình kịch bản được tích hợp vào trong HTML với đặc tính quan trọng là khả năng tạo và sử dụng các đối tượng cho phép người sử dụng xây dựng các trang Web có tính tương tác cao. Ngoài những ưu điểm nổi bật như tính linh hoạt trong lập trình, khả năng chạy và tương thích trên nhiều trình duyệt, khả năng tạo ra các giao diện người dùng tinh tế và phức tạp, cho phép kiểm tra tính hợp lệ của dữ liệu trên máy khách, kích thước các script nhỏ, tốc độ tải xuống nhanh, được hỗ trợ bởi số lượng lớn cộng đồng các nhà phát triển, …, JavaScript cũng có những nhược điểm như khả năng bảo mật kém và hiệu ứng kịch bản phụ thuộc vào trình duyệt và khả năng xảy ra lỗi không mong muốn khi thực thi các script rất cao. Chính tính linh hoạt của JavaScript là một trong những nguyên nhân làm cho lập trình viên dễ gây ra những lỗi không mong muốn trong khi viết chương trình, các đoạn chương trình này được gọi là là các mẫu mã xấu (code smells).

Để nâng cao chất lượng mã nguồn, nhiều nhà khoa học đã nghiên cứu cách thức phát hiện các mẫu mã xấu và đề xuất các phương pháp cải tiến để loại bỏ các đoạn mã xấu. Tuy nhiên, hầu hết các nghiên cứu đều tập trung vào các mẫu mã xấu trong việc xử lý logic chính của chương trình (M. Fowler, K. Beck, J. Brant, W. Opdyke và D. Roberts, 1999[[1]](#footnote-1); R. C. Martin, 2008[[2]](#footnote-2)), việc nghiên cứu về mã xấu trong xử lý ngoại lệ còn rất hạn chế. Trong số ít các nghiên cứu này, R. C. Martin đã chỉ ra một số kỹ thuật để viết mã tốt cho việc xử lý ngoại lệ như sử dụng ngoại lệ (exception) thay vì giá trị trả về (return code), dùng ngoại lệ không kiểm tra (unchecked exception), không trả về null. Những nghiên cứu hiện tại này đều tập trung vào xử lý ngoại lệ trong các ngôn ngữ hướng đối tượng tĩnh (static object oriented languages) như là Java hoặc C#. Việc phát hiện mã xử lý ngoại lệ xấu và cải tiến chúng trong các ngôn ngữ động (dynamic language) như JavaScript là một lĩnh vực mới ở thời điểm hiện tại.

Bất cứ một lập trình viên nào cũng muốn ứng dụng của mình thực thi một cách tốt đẹp theo các yêu cầu đã được đặt ra, hạn chế tối đa những lỗi xảy ra trong lúc thực thi chương trình. Nghĩa là phải xây dựng được một hệ thống mã nguồn tốt, ổn định, tránh những mẫu mã xấu trong lúc viết chương trình. Điều này rất khó thực hiện ngay cả với những lập trình viên có nhiều kinh nghiệm khi xây dựng ứng dụng bằng ngôn ngữ lập trình hướng đối tượng tĩnh. Điều này đặc biệt đúng trong việc xây dựng các ứng dụng với ngôn ngữ lập trình JavaScript - một ngôn ngữ lập trình động, thông dịch, hỗ trợ hướng đối tượng.

Với những vấn đề trình bày ở trên chúng tôi đề xuất đề tài “**Phát hiện và cải thiện mã nguồn xử lý ngoại lệ xấu trong JavaScript**” để nghiên cứu với mục đích giúp cho lập trình viên khi phát triển các ứng dụng phát hiện và tránh những mẫu mã xấu trong xử lý ngoại lệ nhằm giảm thiểu tối đa những lỗi xảy ra trong lúc thực thi chương trình, cụ thể là các lỗi xử lý ngoại lệ đối với ngôn ngữ JavaScript.

# Mục tiêu:

* Nghiên cứu và đưa ra các mẫu mã xấu trong xử lý ngoại lệ đối với ngôn ngữ JavaScript.
* Nghiên cứu những ảnh hưởng của các mẫu mã xấu tới chất lượng mã nguồn, chất lượng sản phẩm.
* Đề xuất các phương thức cải tiến cho mỗi mẫu mã xấu nhằm loại bỏ chúng.
* Đánh giá khả năng áp dụng vào các ngôn ngữ khác.

# Nội dung chính

* Kiến thức cơ sở
  + Cơ chế xử lý ngoại lệ trong các ngôn ngữ lập trình hướng đối tượng
  + Cơ chế xử lý ngoại lệ trong JavaScript
  + Các tính năng chuyên biệt của ngôn ngữ JavaScript trong xử lý ngoại lệ
  + Xử lý ngoại lệ trong lập trình bất đồng bộ JavaScript
  + Các mức độ chắc chắn trong xử lý ngoại lệ
* Các mẫu mã xử lý ngoại lệ xấu trong JavaScript
  + Định nghĩa mẫu mã xử lý ngoại lệ xấu
  + Mô tả mẫu mã xử lý ngoại lệ xấu
  + Áp dụng và phát hiện mẫu mã xử lý ngoại lệ xấu
  + Ví dụ minh họa
* Cách thức cải tiến mã nguồn
  + Thể hiện cải tiến của mẫu
  + Mô hình hóa cách thức cải tiến
  + Những lợi ích mang lại.
  + Cơ chế áp dụng cải tiến
  + Ví dụ, đưa ra ví dụ cụ thể để minh chứng
* Phân tích
  + Phân tích mẫu mã xấu và cách thức cải tiến
  + So sánh mức độ tương đồng với các ngôn ngữ khác.
  + Đánh giá khả năng áp dụng của mỗi cách thức cải tiến vào các ngôn ngữ khác.
  + Thu thập kết quả áp dụng trên các dự án JavaScript mã nguồn mở
* Kết luận và mở rộng
  + Kết luận về kết quả thu được của nghiên cứu
  + Đưa ra các hướng nghiên cứu mở rộng

# Sản phẩm và kết quả dự kiến:

## Sản phẩm khoa học:

- Số bài báo khoa học đăng tạp chí quốc tế: 01

- Số bài báo khoa học đăng tạp chí trong nước: 01

- Số lượng sách xuất bản: 01

## Sản phẩm đào tạo:

- Cử nhân: 01

- Thạc sĩ: 01

## Sản phẩm ứng dụng: Hướng dẫn và các chương trình mẫu áp dụng trong việc xây dựng các ứng dụng.

## Sản phẩm khác: Báo cáo phân tích, tổng kết.

# Hiệu quả dự kiến:

- Kết quả của đề tài là tài liệu tham khảo hữu ích cho những người làm công tác giảng dạy các ngôn ngữ lập trình, những người nghiên cứu về lập trình ứng dụng, sinh viên, học viên cao học và các lập trình viên phát triển các ứng dụng.

- Báo cáo tổng kết đề tài làm tài liệu tham khảo cho sinh viên đại học, học viên cao học chuyên ngành công nghệ phần mềm.

# Nhu cầu kinh phí dự kiến: 450.000.000 đồng (*Bốn trăm năm mươi triệu đồng chẵn*)

# Thời gian nghiên cứu dự kiến: 2017-2018

*Ngày 05 tháng 3 năm 2016*

**Tổ chức, cá nhân đề xuất**

**Trần Thanh Lương**

1. M. Fowler, K. Beck, J. Brant, W. Opdyke and D. Roberts, Refactoring: Improving the Design of Existing Code, Addison-Wesley Professional, July 8, 1999. [↑](#footnote-ref-1)
2. R. C. Martin, Clean Code: A Handbook of Agile Software Craftsmanship, Prentice Hall, August, 2008. [↑](#footnote-ref-2)