**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA KỸ THUẬT MÁY TÍNH**



**BÁO CÁO THỰC TẬP**

**TÌM HIỂU NGHIÊN CỨU VỀ UNIT TEST VÀ INTEGRATION TEST TRONG PHÁT TRIỂN AUTOSAR**

GVHD: Ngô Hiếu Trường

SV thực hiện: Trần Tiến Thiệu

MSSV: 17521084

Lớp: KTMT2017

*Tp.Hồ Chí Minh, tháng 05 năm 2021*

LỜI CẢM ƠN

Trải qua 13 tuần thực tập tại Công ty TNHH phần mềm FPT, từ ngày 01 tháng 03 năm 2021 đến ngày 31 tháng 05 năm 2021, mặc dù thời gian thực tập không quá dài nhưng qua đó em có thể học hỏi cho bản thân mình nhiều kinh nghiệm, kiến thức từ các anh chị. Hơn hết là học hỏi được tác phong, phong cách làm việc trong môi trường công nghiệp, môi trường làm việc tập thể và đó cũng là điều kiện cho em vận dụng những lý thuyết được học tập ở trường để áp dụng vào trong thực tế.

Em xin gửi lời cảm ơn chân thành đến toàn thể quý thầy cô trường Đại học Công nghệ Thông tin - Đại học Quốc gia Hồ Chí Minh, đặc biệt là quý thầy cô trong Khoa Kỹ thuật Máy tính đã giảng dạy, hỗ trợ và giúp đỡ em để có thể được như hiện tại. Cảm ơn thầy Ngô Hiếu Trường đã hỗ trợ, hướng dẫn để em có thể hoàn thành tốt kỳ thực tập này. Những kiến thức quý báu ấy không chỉ được áp dụng ở thời điểm hiện tại mà nó sẽ là hành trang để em có thể tiếp tục sau này.

Em xin chân thành cảm ơn đến quý Công ty TNHH phần mềm FPT đã tạo cơ hội cho em được tiếp sức, thực tập một cách bài bản nhất về những kiến thức em đã được học để vận dụng vào thực tế. Hơn hết, em xin cảm ơn sự hỗ trợ nhiệt tình của anh chị trong dự án DENSO US đã giúp đỡ em không chỉ hoàn thành tốt kỳ thực tập mà còn cho em những kiến thức, kỹ năng cần thiết trong công việc.

Vì kiến thức bản thân còn hạn chế và là lần đầu tiên làm việc trong môi trường công nghiệp nên em không tránh khỏi những sai sót, kính mong nhận được những ý kiến đóng góp từ thầy cũng như quý anh chị trong công ty để em có thể hoàn thiện hơn bản thân mình.

Một lần nữa em xin chân thành cảm ơn!

MỤC LỤC

[CHƯƠNG 1 GIỚI THIỆU VỀ CÔNG TY 1](#_Toc75822559)

[1.1 Tập đoàn FPT 1](#_Toc75822560)

[1.2 Công ty FPT Software 2](#_Toc75822561)

[1.3 Bộ phận thực tập và dự án Denso US 3](#_Toc75822562)

[1.4 Cá nhân tham gia thực tập 3](#_Toc75822563)

[1.5 Lịch làm việc nơi thực tập 3](#_Toc75822564)

[1.5.1 Thời gian thực tập 3](#_Toc75822565)

[1.5.2 Công việc được giao 4](#_Toc75822566)

[1.6 Chính sách bảo mật 4](#_Toc75822567)

[CHƯƠNG 2 Nội dung thực tập 5](#_Toc75822568)

[2.1 Kiến trúc Autosar 5](#_Toc75822569)

[2.1.1 Tổng quan về Autosar 5](#_Toc75822570)

[2.1.2 Nhiệm vụ của các tầng 6](#_Toc75822571)

[2.1.3 Chi tiết về tầng BSW 7](#_Toc75822572)

[2.2 Mô hình Verification and Validation 10](#_Toc75822573)

[2.2.1 Khái niệm mô hình V-model 10](#_Toc75822574)

[2.2.2 Verification Phases 10](#_Toc75822575)

[2.2.3 Validation Phases 11](#_Toc75822576)

[2.3 Cách thức kiểm thử UT, IT 12](#_Toc75822577)

[2.3.1 Tổng quan về Unit Test 12](#_Toc75822578)

[2.3.2 Tổng quan về Integration Test 14](#_Toc75822579)

[2.3.3 Sử dụng Cantata để chạy UT và IT 15](#_Toc75822580)

[CHƯƠNG 3 KẾT QUẢ 19](#_Toc75822581)

[3.1 Về kiến thức 19](#_Toc75822582)

[3.2 Về kỹ năng 19](#_Toc75822583)

[3.3 Kinh nghiệm tích lũy 19](#_Toc75822584)

[CHƯƠNG 4 KẾT LUẬN 20](#_Toc75822585)

[4.1 Kết luận quá trình thực tập của sinh viên 20](#_Toc75822586)

[4.2 Nhận xét, đóng góp ý kiến về quy trình đào tạo của khoa sau khi sinh viên thực tập tại doanh nghiệp 20](#_Toc75822587)

[TÀI LIỆU THAM KHẢO 21](#_Toc75822588)

DANH MỤC HÌNH ẢNH

[Hình 1.1: Trụ sở tập đoàn FPT tại thành phố Hà Nội 1](#_Toc76072024)

[Hình 1.2: Tòa nhà F-Town 1 và F-Town 2 tại Quận 9, TP. Hồ Chí Minh 2](#_Toc76072025)

[Hình 2.1: Logo của chuẩn Autosar 5](#_Toc76072026)

[Hình 2.2:Tầm nhìn của Autosar 5](#_Toc76072027)

[Hình 2.3: AUTOSAR layer 6](#_Toc76072028)

[Hình 2.4:Chi tiết các tầng của AUTOSAR 7](#_Toc76072029)

[Hình 2.5:AUTOSAR MCAL layer 8](#_Toc76072030)

[Hình 2.6: ECU Abstraction layer 8](#_Toc76072031)

[Hình 2.7: Autosar MCAL lớp services 9](#_Toc76072032)

[Hình 2.8: Autosar Layer lớp Complex Drivers 9](#_Toc76072033)

[Hình 2.9: Verification and Validation model 10](#_Toc76072034)

[Hình 2.10: Các tầng kiểm thử của một hệ thống 12](#_Toc76072035)

[Hình 2.11: Logo Cantata 15](#_Toc76072036)

[Hình 2.12: Giao diện tạo một project mới trên Cantata 16](#_Toc76072037)

[Hình 2.13: Test case được chạy thành công 16](#_Toc76072038)

[Hình 2.14: Test case chạy bị lỗi 17](#_Toc76072039)

[Hình 2.15: Minh họa cho test spec UT 18](#_Toc76072040)

[Hình 2.16: Minh họa cho test spec IT 18](#_Toc76072041)

DANH MỤC BẢNG

[Bảng 1.1: Lịch làm việc, học tập 3](#_Toc75810990)

[Bảng 1.2: Nội dung công việc trong quá trình thực tập 4](#_Toc75810991)

[Bảng 2.1: Bảng mô tả điều kiện trong MC/DC 14](#_Toc75810992)

DANH MỤC TỪ VIẾT TẮT

UT - Unit Test

IT - Integration Test

BSW - Basic Software layer

OEM - Original Equipment Manufacturer

API - Application Programing Interface

ECU - Electronic Control Unit

MCAL - Microcontroller Abstraction Layer

RTE - Runtime Environment layer

MCU - Microcontroller Unit

E/E - Exchangeability / Exchangeability

SDLC - Software Development Life Cycle

HLD - High Level Design

SWS - Software Specification

SRS - Software Requirement Specification

GAM - Global Automotive & Manufacturing

TÓM TẮT

“Báo cáo thực tập” là quyển báo cáo tổng kết quá trình thực tập trong thời gian 13 tuần thực tập tại Công ty TNHH phần mềm FPT (FPT software) bắt đầu từ ngày 01 tháng 3 năm 2021 đến hết ngày 31 tháng 05 năm 2021.

Báo cáo trình bày về những nhiệm vụ được giao và hoàn thành, bên cạnh đó nội dung báo cáo còn thể hiện kết quả đánh giá khả năng của sinh viên trong quá trình làm việc tại công ty, cách sinh viên vận dụng những kiến thức được học ở trường để áp dụng vào thực tế.

Báo cáo gồm có 04 chương với các nội dung như sau:

* Chương 1: Cung cấp những thông tin chung về nơi thực tập.
* Chương 2: Cung cấp thông tin về nội dung thực tập
* Chương 3: Kết quả và đúc kết kinh nghiệm sau khi hoàn thành nhiệm vụ được giao.
* Chương 4: Kết luận sau quá trình thực tập tại công ty.

# GIỚI THIỆU VỀ CÔNG TY

## Tập đoàn FPT

FPT (Tập đoàn FPT, tiếng Anh: FPT Group), tên viết tắt của Công ty cổ phần FPT (tên cũ là Công ty Phát triển và Đầu tư Công nghệ), là một trong những công ty dịch vụ công nghệ thông tin lớn nhất tại Việt Nam với lĩnh vực kinh doanh chính là cung cấp các sản phẩm dịch vụ công nghệ thông tin.



Hình 1.1: Trụ sở tập đoàn FPT tại thành phố Hà Nội

FPT có hệ thống 46 văn phòng tại 22 quốc gia trên thế giới, và hạ tầng viễn thông phủ khắp 59/63 tỉnh, thành phố tại Việt Nam. Hình 1.1 là hình ảnh trụ sở chính của tập đoàn FPT tại số 17 Phố Duy Tân, phường Dịch Vọng Hậu, quận Cầu Giấy, Hà Nội, với 7 công ty thành viên và 4 công ty liên kết.

## Công ty FPT Software

Fpt Software là tên gọi khác của công ty TNHH Phần mềm FPT với nhiệm vụ chính là gia công phần mềm tại Việt Nam và nước ngoài.

Thành lập từ năm 1988 đến nay với 3 Trụ sở chính FPT Software đặt tại Việt Nam và một số nước trên toàn thế giới như Hoa Kỳ, Nhật Bản, Malaysia, Đức, Úc, Singapore, Malaysia, Thái Lan và Philippines. Hiện FPT Software đáp ứng nhu cầu gia công phần mềm lớn cho hơn 150 công ty hàng đầu tại 20 quốc gia lớn nhất trên thế giới hiện nay với các hợp đồng lớn có khi đạt cả 1 triệu USD dành cho một số đối tác lớn như Hitachi, NEOPOST, Petronas, Deutsche Bank, và Unilever với hơn 18000 nhân viên trên toàn thế giới.

FPT Software theo đuổi mục tiêu gia công phần mềm để đáp ứng nhu cầu phát triển CNTT của các hãng phần mềm, các công ty lớn trong nước và hơn thế nữa là xuất khẩu phần mềm trên toàn thế giới, mục đích chính là vươn đến tầm cao mới thông qua công nghệ nhằm nâng cao năng suất lao động.

Đến với FPT Software, nhân viên được phát huy tối đa khả năng sáng tạo, nỗ lực đưa các sáng tạo vào các giải pháp công nghệ thông tin nhằm nắm bắt các cơ hội đổi mới và phát triển nhằm đạt được mục tiêu chung của tập đoàn là OneFPT – Công ty xuất khẩu phần mềm hàng đầu Việt Nam.

Đối với Automotive, FPT Software là công ty Việt Nam đầu tiên phát triển công nghệ này, đã và đang thực hiện dự án từ các khách hàng lớn về lĩnh vực ô tô hàng đầu thế giới, với giá trị hàng triệu đô.



Hình 1.2: Tòa nhà F-Town 1 và F-Town 2 tại Quận 9, TP. Hồ Chí Minh

Hình 1.2 là một trong hai cơ sở tại khu công nghệ cao, quận 9, TP. Hồ Chí Minh nơi sinh viên thực tập.

## Bộ phận thực tập và dự án Denso US

GAM (Global Automotive & Manufacturing) là đơn vị trực thuộc FPT Software chuyên về lĩnh vực Automotive, hướng đến các sản phẩm và giải pháp được ứng dụng rộng rãi trong xe hơi tương lai. GAM phát triển ở bốn mảng chính: Ứng dụng trong ô tô, thiết kế, IT/ERP và kiểm thử phần mềm ô tô.

Trong số 7 lĩnh vực được dự đoán tăng trưởng mạnh của Automotive, GAM tập trung phát triển ứng dụng trong ô tô tại 3 lĩnh vực chính là xe tự hành, hệ thống an toàn và hệ thống thông tin giải trí.

Hiện GAM là đối tác của các hãng xe hơi nổi tiếng trên thế giới đến từ Nhật Bản, Hàn Quốc, EU, … với đội ngũ chuyên gia nhiều năm kinh nghiệm trong lĩnh vực CAD/CAM/CAE và công nghệ ô tô. Tính đến nay, GAM đang tham gia khoảng 150 dự án trong tất cả các lĩnh vực Automotive.

Dự án Denso US là sự hợp tác giữa FPT và đối tác để phát triển về tầng Layer BSW trong kiến trúc Autosar. Đây là dự án với quy mô khoảng 17 thành viên.

## Cá nhân tham gia thực tập

- Họ và tên sinh viên: Trần Tiến Thiệu

- MSSV: 17521084

- Công ty thực tập: FPT Software HCM

- Địa điểm thực tập: Tòa nhà F-town 2, Quận 9

- Chức vụ công việc: Thực tập sinh

## Lịch làm việc nơi thực tập

### **Thời gian thực tập**

|  |  |  |
| --- | --- | --- |
| **Thời gian** | **Thứ** | **Giờ làm việc** |
| 01/03/2021 – 31/05/2021 | Thứ 2, thứ 4, thứ 6 | 8h00 – 17h00 |
| Quản lý chịu trách nhiệm: Lê Nguyễn Quang Vinh  Mail: VinhLNQ1@fsoft.com.vn | | |

Bảng 1.1: Lịch làm việc, học tập

### **Công việc được giao**

|  |  |
| --- | --- |
| Giai đoạn | Nhiệm vụ |
| Tuần 1  1/3 – 5/3 | * Tham gia khóa training dayone dành cho người mới giới thiệu về công ty cũng như cách thức an toàn bảo mật thông tin * Giới thiệu về các tool hay dùng trong công ty như git, svn, ... * Giới thiệu và đọc tài liệu về Autosar |
| Tuần 2  8/3 – 12/3 | * Giới thiệu về UT * Giới thiệu và đọc tài liệu về CANTATA và cách sử dụng để chạy UT |
| Tuần 3 – 5  15/3 – 2/4 | * Sử dụng CANTATA để thực hành UT cho các module * Viết báo cáo sau khi chạy UT |
| Tuần 6  5/4 – 9/4 | * Giới thiệu về IT * Giới thiệu cách thức sử dụng CANTATA để chạy IT và cách sửa lỗi khi chạy IT |
| Tuần 7 – 12  12/4 – 21/5 | * Sử dụng CANTATA để thực hành IT cho các module * Viết báo cáo sau khi chạy IT * Tham gia các khóa học online trên hệ thống công ty: điện toán đám mây, các công cụ mà các nhân viên khi làm việc cần biết đến, kỹ năng cần thiết cho lập trình, ... |
| Tuần 13  24/5 – 28/5 | * Nhận xét, góp ý và bàn giao công việc với Project Manager |

Bảng 1.2: Nội dung công việc trong quá trình thực tập

## Chính sách bảo mật

Vì là một công ty làm việc trong lĩnh vực công nghệ nên việc bảo mật đối với công ty rất quan trọng. Những thông tin tài liệu liên quan đến bí mật của công ty không thể được tự ý đem về nhà cũng như tuyệt đối không được cho người ngoài công ty được biết.

Theo chính sách bảo mật của công ty, sinh viên chỉ có thể mô tả sơ lược về các công việc mà mình đã làm chứ không thể đi sâu vào chi tiết.

# Nội dung thực tập

## Kiến trúc Autosar

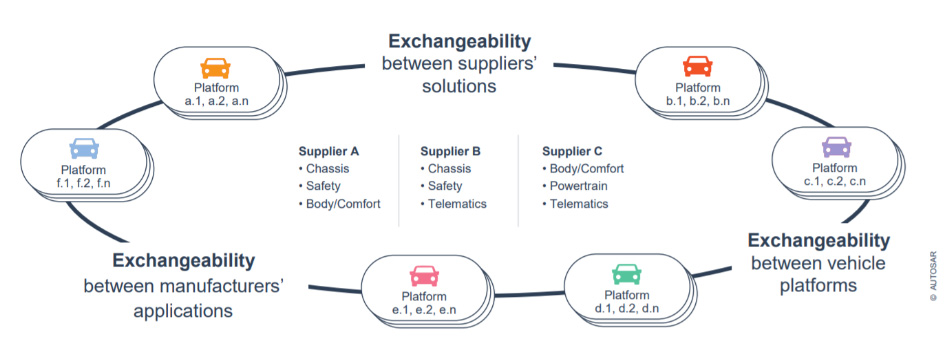
### **Tổng quan về Autosar**

* AUTOSAR [1] là kiến trúc chung dành riêng cho ECU xe hơi, một quy chuẩn chung dành cho phát triển xe hơi và các thiết bị bổ trợ cho xe hơi đề ra để các thiết bị trong chuẩn có thể dễ dàng tương thích với nhau bỏ qua khác biệt phần cứng và nhà sản xuất. Logo của Autosar được biết đến như hình 2.1.



Hình 2.1: Logo của chuẩn Autosar

* AUTOSAR được thành lập nhằm mục đích cải thiện khả năng quản lý độ phức tạp của kiến trúc E/E tích hợp thông qua việc tăng khả năng tái sử dụng và trao đổi mô-đun phần mềm giữa OEMs và nhà cung cấp.



Hình 2.2: Tầm nhìn của Autosar

* Tiêu chuẩn hóa kiến trúc phần mềm của ECUs và mở đường cho các hệ thống điện tử cải tiến nhằm cải thiện hiệu suất, an toàn và bảo mật. Phần cứng và phần mềm sẽ được phát triển độc lập, giảm thời gian và chi phí phát triển. Việc sử dụng lại phần mềm giúp các nhà phát triển tăng chất lượng và hiệu quả.
* Tổng thể thì AUTOSAR có 4 tầng chính:
* Application layer
* Runtime Eviroment layer
* Basic sofware layer
* Microcontroller layer
* Hình 2.3 miêu tả các lớp từ trên xuống dưới của chuẩn AUTOSAR.

Graphical user interface, chart, website, treemap chart

Description automatically generated

Hình 2.3: AUTOSAR layer

### **Nhiệm vụ của các tầng**

* **Microcontroller layer:**
* Là lớp phần cứng trong sơ đồ như vi điều khiển hoặc vi xử lý và các thành phần đi kèm.
* Ứng với mỗi microcontroller sẽ là một bộ phần mềm AUTOSAR riêng biệt.
* **Basic Software layer (BSW):**
* Đây là tầng có nhiệm vụ triển khai các [application programming interfaces](https://en.wikipedia.org/wiki/Application_programming_interface) (APIs) theo tiêu chuẩn AUTOSAR, dựa Software Specification (SWS).
* Mỗi dòng microcontroller khác nhau sẽ cần một bộ BSW riêng biệt.
* Là tầng trung gian giữa các tầng trên và tầng microcontroller nên nó sẽ đóng vai trò là tầng thực hiện tính trừu tượng hóa của AUTOSAR.
* Đối với các tầng trên tầng này, mọi thiết bị theo chuẩn AUTOSAR đều có thể dùng chung phần mềm với nhau mà không bị ảnh hưởng.
* **Runtime Environment layer (RTE):**
* Tầng này là tầng liên kết giữa BSW và Application, là tầng phân chia nhiệm vụ giữa một kỹ sư lập trình nhúng và một kỹ sư lập trình ứng dụng.
* Tầng này sẽ đóng vai trò trừu tượng hóa hoàn toàn các tác vụ xử lý nội bộ phía dưới và là một interface cho phép các lập trình viên ứng dụng dễ dàng triển khai ứng dụng mà không cần có các kiến thức liên quan về phần cứng.
* Application layer:
* Tầng này là tầng cuối cùng, cao nhất trong AUTOSAR.
* Hoàn toàn độc lập với ECU, có thể được lập trình bời bất kỳ lập trình viên ứng dụng nào mà không cần phải hiểu về phần cứng.

### **Chi tiết về tầng BSW**

Graphical user interface, website

Description automatically generated

Hình 2.4:Chi tiết các tầng của AUTOSAR

* Tương tự với cấu trúc chung, hình 2.4 miêu tả chi tết ở nội bộ tầng này cũng sẽ được sắp lớp cụ thể theo từng thành phần tiếp giáp nhau từ thấp tới cao theo mức độ trừu tượng hóa.
* **Microcontroller Abstraction layer (MCAL)**
* Đây là lớp có sự can thiệp và phụ thuộc vào phần cứng nhiều nhất trong hệ thống. Lớp này có sự hiện diện của các trình điều khiển (driver) điều khiển trực tiếp các thành phần trong microcontroller như là các thanh ghi để tác động lên phần cứng. Hình 2.5 mô tả chi tiết các giao thức trong lớp MCAL:

**Chart

Description automatically generated**

Hình 2.5:AUTOSAR MCAL layer

* Lớp này có sự khác biệt cốt lõi giữa các dòng microcontroller với nhau nên với mỗi dòng microcontroller chúng ta sẽ triển khai riêng lớp này cho nó, không thể dùng chung hay tái sử dụng.
* **ECU Abstraction layer**

Table, calendar

Description automatically generated

Hình 2.6: ECU Abstraction layer

* Theo hình 2.6, chúng ta có thể thấy được lớp này nằm giữa lớp drivers và services nên ở lớp này ta cần trừu tượng hóa phần cứng cơ bản, giản hóa những giao thức phần cứng nội bộ ở lớp dưới để dễ dàng hiểu được với một lập trình viên.
* **System Services**
* Đây là một lớp cung cấp các dịch vụ cụ thể tới cho hệ thống – ví dụ như hệ điều hành thời gian thực (AUTOSAR RTOS).

Table, calendar

Description automatically generated

Hình 2.7: Autosar MCAL lớp services

* Như đã thấy ở trên hình 2.7, ở lớp này chúng ta vừa tiếp giáp với tầng RTE tức phải có đầy đủ tính trừu tượng, nhưng cũng có một phần nhỏ tiếp giáp với tầng MCAL để có thể điều khiển trực tiếp một số thành phần của phần cứng khi cần thiết.
* Sau lớp này, tầng trên sẽ sử dụng khái niệm services, sẽ là sự tương tác giữa nhà cung cấp services và thành phần sử dụng services được cấp phát bởi RTE.
* **Complex Drivers**

**Table, calendar

Description automatically generated**

Hình 2.8: Autosar Layer lớp Complex Drivers

* Đây là lớp cuối cùng mà mình đề cập tới bởi vì đây là một lớp tự do nối trực tiếp 2 layers với nhau như hình 2.8.
* Ở các lớp khác trong tiêu chuẩn AUTOSAR, ứng với mỗi thành phần trong lớp sẽ có những yêu cầu cụ thể và rõ ràng để đảm bảo tính đồng nhất khi triển khai, nhưng cũng vì vậy mà các nhà sản xuất sẽ bị gò bó bởi chúng mà không thể tạo ra các thành phần riêng biệt của bản thân. Chính vì lẽ đó mà lớp đặc biệt cuối cùng này được ra đời. Đây là một lớp đặc biệt, được AUTOSAR dành quyền thiết kế và sử dụng hoàn toàn cho các nhà cung cấp phần mềm, khiến cho AUTOSAR có được sự đa dạng hơn khi các nhà cung cấp phần mềm đưa ra các tính năng tùy chỉnh riêng biệt của họ.

## Mô hình Verification and Validation

### **Khái niệm mô hình V-model**

* Software Development Life Cycle (SDLC) là một quá trình theo sau cho một dự án phần mềm, trong một tổ chức phần mềm bao gồm một kế hoạch chi tiết mô tả làm thế nào để phát triển, duy trì, thay đổi hoặc nâng cấp phần mềm.



Hình 2.9: Verification and Validation model

* V-model là một SDLC trong đó việc thực thi các quy trình diễn ra tuần tự theo hình chữ V được biết đến với tên gọi là **Verification and Validation model**.
* V-model dựa trên sự liên kết của mỗi giai đoạn thử nghiệm (testing stage) tương ứng cho mỗi giai đoạn phát triển (development stage). Quá phát triển một hệ thống được thể hiện chi tiết trong hình 2.9.

### **Verification Phases**

* **Business Requirement Analysis:** là giai đoạn đầu tiên trong quy trình phát triển phần mềm. Ở giai đoạn này người phát triển (development) sẽ phải nghiên cứu yêu cầu khách hàng. Từ đó lập ra bản thảo chi tiết các công việc sẽ làm, kế hoạch đặt ra để thực hiện theo đúng yêu cầu.
* **System Design:** Khi đã nắm rõ yêu cầu thì sẽ tiến hành đến quy trình thiết kế hệ thống. Giai đoạn này sẽ dựa vào yêu cầu của khách hàng đồng thời cũng phải dựa vào đặc tính của phần cứng hỗ trợ mà định nghĩa ra các thành phần khác nhau trong hệ thống.
* **Architectural Design:** Ở giai đoạn này chúng ta sẽ phải định nghĩa được tên các hàm, kiểu dữ liệu, cấu trúc, cấu tạo và tổ chức files. Giai đoạn này còn được biết đến với tên gọi High Level Design (HLD).
* **Module design (Detailed Design):** Giai đoạn này quyết định cách thức làm việc của coding vì nó chứa các flowchart quy định luồng xử lý. Đồng thời ở giai đoạn này còn quy định về giá trị trả về, phương thức thông báo có lỗi, …
* **Coding:** là giai đoạn cuối của việc tạo ra sản phẩm (chưa qua kiểm duyệt). Người phát triển sẽ phải dựa vào Detailed Design mà viết ra các firmware, APIs tương ứng tuân thủ các flowchart trước đó đã thiết kế.

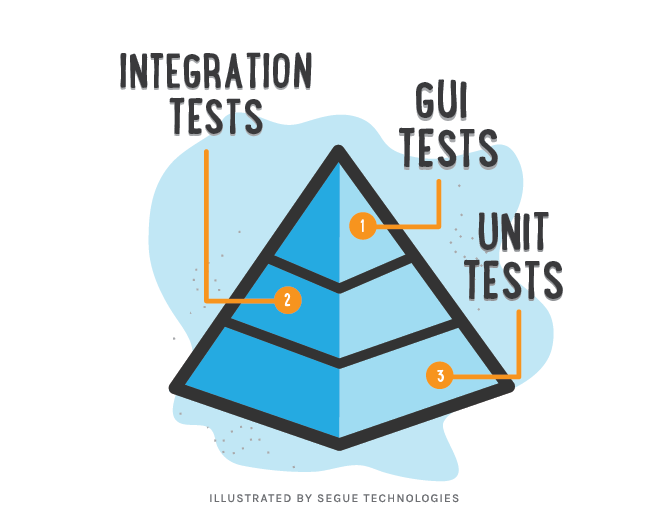
### **Validation Phases**

* **Unit testing:** là đánh giá mức độ bao phủ các trường hợp của source code. Một source code hoàn chỉnh phải có độ phủ gần như 100% cho tất cả các nhánh. Một input phải tương ứng với output mong đợi. Đây là functional testing, chỉ tập trung vào kiểm tra khả năng của hàm trong các trường hợp lý tưởng.
* **Integration testing:** Ở bước này tester phải kiểm tra chức năng các APIs. Người kiểm duyệt phải hiểu được input và output của các APIs từ đó viết ra các test case kiểm tra kết quả. Ở bước này cho ra đánh giá cụ thể khả năng đáp ứng của module và khả năng đáp ứng các yêu cầu đề ra trong SRS của module.
* **System testing:** Ở bước này sẽ kết hợp nhiều module lại với nhau để đảm bảo rằng các resource không bị tranh chấp. Bên cạnh đó yếu tố performance và memory resource cũng sẽ được kiểm tra đánh giá.
* **Acceptance testing:** đây là bước kiểm tra cuối cùng trước khi giao sản phẩm cho khách hàng. Ở bước này cần kiểm tra chức năng của sản phẩm có đáp ứng được yêu cầu khách hàng hay không. Độ tin cậy của sản phẩm được đánh giá dựa vào các test case có được kiểm thử với tất cả trường hợp có thể xảy ra hay không. Bước này được thực hiện với những trường hợp cụ thể do bên khách hàng kiểm thử và báo kết quả.

## Cách thức kiểm thử UT, IT

### **Tổng quan về Unit Test**

* Unit Testing (UT) là một loại kiểm thử phần mềm trong đó các đơn vị hay thành phần riêng lẻ của phần mềm được kiểm thử. Kiểm thử đơn vị được thực hiện trong quá trình phát triển ứng dụng. Mục tiêu của kiểm thử đơn vị là cô lập một phần code và xác minh tính chính xác của đơn vị đó.



Hình 2.10: Các tầng kiểm thử của một hệ thống

* Một Unit là một thành phần PM nhỏ nhất mà ta có thể kiểm tra được như các hàm (Function), thủ tục (Procedure), lớp (Class), hoặc các phương thức (Method).
* Công việc viết Unit Test có thể đem lại những lợi ích sau [2][3]:
* Đảm bảo chất lượng của source code.
* Tìm kiếm và phát hiện các vấn đề tìm ẩn có thể gây ảnh hưởng không tốt đến chương trình
* Giảm chi phí sửa chữa và chi phí kiểm soát chất lượng dự án
* Quá trình phát triển một sản phẩm phần mềm luôn đi cùng với các lỗi phát sinh, nhiệm vụ của các nhà phát triển là phải hạn chế số lượng lỗi và hạn chế chi phí sửa lỗi, lỗi được phát hiện càng sớm thì chi phí sửa lỗi càng ít và ngược lại.
* Test coverage được định nghĩa là một kỹ thuật xác định xem các trường hợp thử nghiệm có thực sự bao trùm mã ứng dụng hay không và bao nhiêu mã được thực hiện khi chạy các trường hợp thử nghiệm đó, có các kỹ thuật Test Coverage phổ biến như sau [4]:
* **C0**: Tỉ lệ bao phủ câu lệnh (Command Coverage Ratio hay Statement Coverage) đảm bảo rằng tất cả các dòng lệnh trong mã nguồn đã được kiểm tra ít nhất một lần. Nó cung cấp các chi tiết của cả hai khối mã được thực thi và thất bại trong tổng số các khối mã.
* **C1**: Tỉ lệ bao phủ nhánh (Branch Coverage). Các nhà phát triển không thể viết mã trong một chế độ liên tục, tại bất kỳ điểm nào họ cần phân nhánh mã để đáp ứng các yêu cầu chức năng. Sự phân nhánh trong mã thực sự là một bước nhảy từ điểm quyết định này sang điểm khác. Branch Coverage kiểm tra mọi đường dẫn có thể hoặc chi nhánh trong mã được kiểm thử.
* **C2**: Tỉ lệ bao phủ điều kiện (Condition Coverage) tập trung vào điều kiện trong nhánh điều kiện, bao phủ test toàn bộ các kết quả điều kiện. Điểm khác với bao phủ nhánh (C1) là: Cho dù nhiều điều kiện bị ràng buộc với nhau bằng AND hay OR, thì vẫn sẽ được coi như các điều kiện độc lập với nhau.
* **MC/DC** (Modified Condition Decision Coverage) là chuẩn đo source về độ bao phủ. Kết quả trả về sau khi kiểm tra xong là True hoặc False, cùng với 1 thông số quan trọng nhất là % độ bao phủ của hàm đó.
* Transaction Level Modeling (TLM) là một phần của SystemC. TLM được sử dụng để mô hình hóa mức Transaction, tập trung vào việc mô hình cách thông tin giữa các process thông qua việc gọi và sử dụng các chức năng có sẵn. Dưới góc nhìn phần cứng, TLM sẽ được sử dụng để mô hình các trao đổi thông tin giữa các khối (block) hoặc module thay vì dùng cách kết nối từng tín hiệu.
* Có ít nhất một test case cho ra kết quả là TRUE nhờ điều kiện TRUE.
* Có ít nhất một test case cho ra kết quả là FALSE nhờ điều kiện FALSE.
* Cách tạo test case MC/DC dựa theo bảng 2.1:
* Xác định giá trị trung lập và giá trị quyết định kết quả.
* Giá trị quyết định kết quả là giá trị sẽ quyết định kết quả là true hay false, nếu nó là true thì kết quả ra là True, nếu nó là false kết quả ra là False.
* Giá trị trung lập là giá trị không ảnh hưởng đến kết quả.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | A | B | C | (A & B) | C |
| 1 | T | T | T | T |
| 2 | T | T | F | T |
| 3 | T | F | T | T |
| 4 | T | F | F | F |
| 5 | F | T | T | T |
| 6 | F | T | F | F |
| 7 | F | F | T | T |
| 8 | F | F | F | F |

Bảng 2.1: Bảng mô tả điều kiện trong MC/DC

### **Tổng quan về Integration Test**

* Kiểm thử tích hợp (Integration testing) [5] là một giai đoạn trong kiểm thử phần mềm. Mỗi module phần mềm riêng biệt được kết hợp lại và kiểm thử theo nhóm.
* Kiểm thử tích hợp xảy ra sau kiểm thử đơn vị (Unit Test) và trước kiểm thử xác nhận. Kiểm thử tích hợp nhận các module đầu vào đã được kiểm thử đơn vị, nhóm chúng vào các tập hợp lớn hơn, áp dụng các ca kiểm thử đã được định nghĩa trong kế hoạch kiểm thử tích hợp vào tập hợp đó, và cung cấp đầu ra cho hệ thống tích hợp.
* Mặc dù mỗi module đều được kiểm thử đơn vị (Unit test) nhưng vẫn có thể tồn tại lỗi do các nguyên nhân sau:
* Một Module nói chung được thiết kế bởi một lập trình viên có hiểu biết và logic lập trình có thể khác với các lập trình viên khác. Kiểm thử tích hợp là cần thiết để đảm bảo tính hợp nhất của phần mềm.
* Tại thời điểm phát triển module vẫn có thể có thay đổi trong spec của khách hàng, những thay đổi này có thể không được kiểm tra ở giai đoạn unit test trước đó.
* Giao diện và cơ sở dữ liệu của các module có thể chưa hoàn chỉnh khi được ghép lại.
* Khi tích hợp hệ thống các module có thể không tương thích với cấu hình chung của hệ thống.
* Thiếu các xử lý ngoại lệ có thể xảy ra.
* Các bước tạo test case IT:
* Chuẩn bị Integration Test Plan dựa trên SRS.
* Thiết kế các kịch bản thử nghiệm, trường hợp, và script.
* Thực hiện kiểm tra theo test case đã viết.
* Theo dõi và tái kiểm tra các lỗi ở trên.

### **Sử dụng Cantata để chạy UT và IT**

* Cantata (hình 2.11) là một đơn vị và công cụ kiểm tra tích hợp, cho phép các nhà phát triển xác minh đoạn mã tiêu chuẩn hoặc đoạn mã quan trọng của doanh nghiệp trên các nền tảng gốc và nhúng của máy chủ.



Hình 2.11: Logo Cantata

* Quá trình testing bao gồm các bước sau [6]:
* Phân tích diagram design và điền giá trị trả về mong muốn và giá trị truyền vào vào trong test spec.
* Diagram design là sơ đồ từ detailed spec để coder có thể dựa vào và viết các dòng lệnh, kiểm thử xem các đoạn lệnh có giống như mô tả hay không.
* Từ test spec sử dụng công cụ do đội ngũ nhân viên công ty viết ra để xuất ra file chứa các test case.
* Tất cả các test case để kiểm thử cho cùng một đoạn lệnh sẽ được xuất ra trong một file để dễ dàng quản lý cũng như nhập vào Cantata.
* Khởi tạo một project mới trên Cantata (hình 2.12), truyền mã cần test cùng với các test case. Chạy các đoạn mã cùng test case, Cantata sẽ trả về kết quả test case đó được thông qua hay có bất kỳ lỗi nào không.

Graphical user interface, application

Description automatically generated

Hình 2.12: Giao diện tạo một project mới trên Cantata

* Nếu các test case được chạy thành công (hình 2.13), tester tổng hợp các config lại với nhau và sử dụng công cụ để tạo báo cáo.

Table

Description automatically generated

Hình 2.13: Test case được chạy thành công

* Nếu các test case bị lỗi (hình 2.14) hoặc trong báo cáo trả về thiếu trường hợp kiểm tra, tester quay lại một trong các bước trên để sửa lỗi hoặc thêm test case mới.

Table

Description automatically generated

Hình 2.14: Test case chạy bị lỗi

* Để biết xảy ra lỗi gì thì tester mở tệp tin mà Cantata đã xuất ra, sau đó đối chiếu các giá trị trong test case.
* Để biết cần tạo mới test case nào thì tester cần đọc hiểu được lệnh của hàm đang test và theo dõi luồng chạy của chúng để xem bị thiếu ở đâu.
* Sau khi thêm test case để hoàn tất kiểm thử đoạn lệnh, tester nên thêm vào test spec ngay để những người sau có thể nắm được toàn bộ quá trình và dễ dàng đánh giá lại kết quả.
* Sau khi hoàn tất việc chạy các test case, tester sử dụng công cụ để tạo báo cáo, báo cáo này thể hiện độ bao phủ của việc kiểm thử, nếu không hoàn thiện thì cần xem lại để xác định nguyên nhân và sửa chữa hoặc ghi nhận lại.

Graphical user interface, text, application, email

Description automatically generated

Hình 2.15: Minh họa cho test spec UT

Graphical user interface, text, application, email

Description automatically generated

Hình 2.16: Minh họa cho test spec IT

Mỗi loại test khác nhau yêu cầu các test spec khác nhau, ta có thể thấy mẫu test spec cho UT như hình 2.15 và IT như hình 2.16.

# KẾT QUẢ

## Về kiến thức

* Cải thiện lập trình C/C++.
* Quy trình để phát triển một phần mềm.
* Quy trình kiểm thử (testing).
* Công cụ hỗ trợ phát triển và kiểm thử.
* Cấu trúc phân tầng và kiến trúc hoạt động của AUTOSAR.

## Về kỹ năng

* Nâng cao khả năng đọc và viết bằng tiếng Anh: Vì tất cả văn bản như tài liệu kỹ thuật, báo cáo viết hằng ngày, thông báo từ công ty, email, ... đều viết bằng tiếng Anh. Vì vậy, điều đó đòi hỏi nhân viên phải rèn luyện kỹ năng đọc và viết thật tốt.
* Bên cạnh đó cũng cải thiện được kỹ năng nghe, nói tiếng Anh thông qua các buổi thuyết trình, chia sẻ của mỗi nhân viên về những kiến thức thú vị mà họ có được trong quá trình làm việc và trong cuộc sống.
* Học kỹ năng viết báo cáo, viết email: do tính chất công việc thì mỗi ngày nhân viên đều phải thực hiện công việc này. Từ đó, sinh viên học được cách viết sao cho ngắn gọn, súc tích nhưng vẫn truyền đạt đầy đủ ý.

## Kinh nghiệm tích lũy

* Làm quen và dần thích nghi với môi trường làm việc chuyên nghiệp ở công ty và tác phong làm việc như một kỹ sư thực thụ.
* Cải thiện khả năng làm việc nhóm lẫn khả năng làm việc độc lập.
* Quản lý thời gian: sắp xếp thời gian làm việc một cách hợp lý để luôn đảm bảo kiểm soát được tiến độ công việc và hoàn thành nhiệm vụ đúng thời hạn.

# KẾT LUẬN

## Kết luận quá trình thực tập của sinh viên

Quá trình thực tập ở công ty đã giúp sinh viên áp dụng những kiến thức học được trong công việc, biết được nhiều kiến thức chuyên ngành khác có ích cho công việc sau này. Đồng thời, tác phong làm việc và những yêu cầu dành cho bản thân khi làm việc trong một môi trường chuyên nghiệp là những điều quan trọng khác mà sinh viên đã nhận thức được.

Do thời gian thực hiện báo cáo có hạn cùng quy định về việc tiết lộ thông tin trong hợp đồng với công ty, sinh viên chi giới thiệu các nội dung cơ bản, một số nội dung chi tiết về hoạt động của công ty đã không được trình bày trong bài viết này.

## Nhận xét, đóng góp ý kiến về quy trình đào tạo của khoa sau khi sinh viên thực tập tại doanh nghiệp

Qua 13 tuần thực tập tại công ty TNHH phần mềm FPT sinh viên nhận thấy các môn học được giảng dạy ở Khoa là phù hợp với nhu cầu kiến thức cần có trong việc làm của doanh nghiệp, cách dạy các môn song song giữa 02 chuyên ngành “Thiết kế hệ thống nhúng và Robotics” và “Thiết kế vi mạch” đảm bảo cho sinh viên có đủ kiến thức trong ngành Kỹ thuật Máy tính về tư duy thiết kế các mạch điện tử và lập trình điều khiển tự động các thiết bị.

Sinh viên cũng mong rằng Khoa trang bị thêm kiến thức về việc viết các công cụ phần mềm đơn giản mà hiệu quả để thiết kế và kiểm thử mạch điện tử và hệ thống nhúng nhanh hơn, hiệu quả hơn cũng như giúp sinh viên có thêm kiến thức thực tế bám sát hơn với doanh nghiệp.

Chương trình đào tạo của Khoa là phù hợp với nhu cầu của công ty. Việc thực hiện đồ án tương ứng với mỗi môn học giúp cho sinh viên sẽ không quá bỡ ngỡ với môi trường làm việc thực tế tại công ty. Tuy nhiên sinh viên nhận thấy vẫn còn nhiều bạn bị bỡ ngỡ, chưa có định hướng về việc giải quyết các bài toán đặt ra trong mỗi đồ án được giao. Vì vậy sinh viên có ý kiến là Khoa nên thường xuyên tổ chức những buổi seminar định kỳ để truyền đạt cho những bạn sinh viên năm nhất các kỹ năng phân tích, tìm hướng giải quyết những vấn đề gặp phải trong quá trình làm đồ án của mình.

# TÀI LIỆU THAM KHẢO

1. AUTOSAR, “Classic platform”, Nov 30,2020. [Online]. Available: <https://www.autosar.org/standards/classic-platform/>
2. Sergey Kolodiy, “Unit Tests, How to Write Testable Code and Why it Matters”, 2016. [Online]. Available: <https://www.toptal.com/qa/how-to-write-testable-code-and-why-it-matters>
3. Guru99, “Unit Testing Tutorial: What is, Types, Tools & Test EXAMPLE”, Unknown. [Online]. Available: https://www.guru99.com/unit-testing-guide.html
4. Segue Technologies, “How Much Test Coverage Is Enough For Your Testing Strategy?”, June 13, 2014. [Online]. Available: <https://www.seguetech.com/how-much-test-coverage-enough-testing-strategy/>
5. Guru99, “Integration Testing: What is, Types, Top Down & Bottom Up Example”, Unknown. [Online]. Available: https://www.guru99.com/integration-testing.html
6. QA Systems, “Automated unit & integration testing for C & C++”, Unknown. [Online]. Available: <https://www.qa-systems.com/tools/cantata/>