INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA

FABIO JUNIOR JUBELLI

ANÁLISE DA CERTIFICAÇÃO E ENSAIO DE TESTES DE SOFTWARE APLICADO À SISTEMAS ELETRÔNICOS EMBARCADOS

JOINVILLE

2022

FABIO JUNIOR JUBELLI

ANÁLISE DA CERTIFICAÇÃO E ENSAIO DE TESTES DE SOFTWARE APLICADO À SISTEMAS ELETRÔNICOS EMBARCADOS

Monografia apresentada ao Bacharelado em Engenharia Elétrica do Campus Joinville do Instituto Federal de Santa Catarina para a obtenção do diploma de Bacharel em Engenharia Elétrica

Orientador: Michael Klug, Dr.

Joinville

2022

FABIO JUNIOR JUBELLI

ANÁLISE DA CERTIFICAÇÃO E ENSAIO DE TESTES DE SOFTWARE APLICADO À SISTEMAS ELETRÔNICOS EMBARCADOS

Este trabalho foi julgado adequado para obtenção do título de bacharel em engenharia elétrica pelo Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina, e aprovado na sua forma final pela comissão avaliadora abaixo indicada.

Joinville, 17 de fevereiro de 2022.

_____

Prof. Michael Klug, Dr.

Orientador

Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina

_____

Prof. Rodrigo Coral, Dr.

Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina

_____

Prof. José Flávio Dums, Dr.

Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina

Não apenas este trabalho, mas todas as minhas
conquistas pessoais e profissionais são dedicadas
a Deus e a tudo que ele representa,
à minha família professore e amigos.

AGRADECIMENTOS

Agradeço primeiramente a Deus pelos acontecimentos em minha vida e por permitir que eu pudesse superar as dificuldades que tive durante o desenvolvimento deste trabalho e que me ensinaram que nunca se deve desistir.

Aos meus pais pelo amor, dedicação, educação e confiança, sentimentos que fizeram com que lutasse com determinação.

À minha esposa por toda a compreensão, apoio e ajuda que obtive nos momentos de dificuldades.

Aos meus irmãos, amigos de todas as horas.

Ao meu amigo, Lucas Bezerra de Menezes por me ajudar na ideia inicial desse trabalho e me incentivar a dar continuidade no mesmo.

Aos professores, orientadores e amigos Michael Klug (TCC) e Stefano R. Zeplin (Projeto Integrador III) pelo apoio e compreensão pessoal neste trabalho.

A todos os professores, colegas e amigos que de alguma forma participaram de mais esta etapa que cumpri de minha vida. A todos fica aqui registrado o meu muito OBRIGADO.

RESUMO

Os sistemas eletrônicos possuem grande complexidade, dessa forma exigindo a implementação de metodologias e ferramentas para desenvolvimento do(s) software(s) neles existentes, e para os testes necessários à validação deles. Esses sistemas são muito abrangentes, podendo estar inseridos desde a nanotecnologia até na robustez de maquinário para serviço de mineração, por exemplo. Uma das ramificações dos sistemas eletrônicos são os sistemas embarcados automotivos, onde a grande maioria das funcionalidades é controlada por software, dentro de uma vasta quantidade de ECUs (*Electronic control unit* – unidade de controle eletrônico). Nesse contexto, os testes para software embarcado têm uma ampla área de pesquisa, no qual o principal intuito é o de implementar técnicas que busquem encontrar o maior número de defeitos e/ou falhas ainda nas fases de desenvolvimento. Este trabalho realiza um estudo sobre o processo de certificação e ensaios no desenvolvimento de software aplicado à eletrônica embarcada e busca analisar os métodos de testes para propor uma melhor aplicação para cada situação.

Palavras-Chave: Sistemas Eletrônicos. *Software*. *ECU*. Eletrônica. Veículos.

ABSTRACT

Electronic systems have great complexity, thus requiring the implementation of methodologies and tools for software development in them, and for the tests necessary for their validation. These systems are very comprehensive and can be embedded from nanotechnology to the robustness of machinery for mining services, for example. One of the branches is the automotive embedded systems, where the vast majority of functionality is controlled by software, within a vast number of ECUs (Electronic control unit). In this context, tests for embedded software have a wide area of research, in order to implement techniques that seek to find the greatest number of defects and/or failures still in the development phases. This work conducts a study on the certification and testing process in software development applied to embedded electronics and seeks to analyze the test methods to propose a better application for each situation.

Keywords: Electronic Systems. Software. ECU. Electronics. Vehicles.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AD – Analógico-Digital

ABS - *Anti lock Braking System* (Sistema de travamento antibloqueio)

BCM - *Body Control Module* (Módulo de controle da carroceria)

BDUF - *Big Design Up Front* (Projetar antes)

CAN - *Controller Area Network* (Rede de área do controlador)

CI - Circuito Integrado

CPU - *Central Processing Unit* (Unidade de Processamento Central)

DA - Digital-Anallógico

DLC - *Data Lenght Code* (Código de Comprimento de Dados)

ECU - *Electronic Control Unit* (Unidade de Controle eletrônico)

EOF - *end of frame* (Fim do frame/quadro)

FIBEX - *FlexRay network database* (Banco de dados de rede FlexRay)

GPS - *Global Positioning System* (Sistema de Posicionamento Global)

GUI - *graphical user interface* (interface gráfica para o usuário)

HD - *Hard Disk* (Disco Rígido)

ID - *Identification* (Identificação)

IDE - *Identifier Extension Bit* (Bit de extensão do identificador)

IEC - *International Electrotechnical Commission* (Comissão Eletrotécnica Internacional)

IEEE - *Institute of Electrical and Electronics Engineers* (Instituto de Engenheiros Eletricistas e Eletrônicos)

IFSC - Instituto Federal de Santa Catarina

IHM - interface homem-máquina

IPO - *Input-Processing-Output* (Entrada-Processamento-Saída)

ISO - *International Organization for Standardiation* (Organização Internacional para Normatização)

LDF - *LIN description file* (Arquivo de descrição LIN)

LIN - *Local Interconnect Network* (Rede de interconexão local)

MOST - *Media Oriented System Transport* (Transporte de sistema orientado à mídia)

NCF - *Node capability files* (Arquivos de capacidade do nó)

OBD - *onboard diagnostics* (diagnósticos de bordo)

PC - *Personal Computer* (Computador Pessoal)

PCM - *Powertrain Control Module* (Módulo de controle do trem de força)

QA - *Quality Assurance* (garantia de qualidade)

QC - *Quality Control* (controle de qualidade)

RAM - *Random Access Memory* (Memória Estática de Acesso Aleatório)

ROM - *Read Only Memory* (Memória Somente Leitura)

SAE - *Society of Automotive Engineers* (Sociedade de Engenheiros Automotivos)

SOF - *start of frame* (Início do frame/quadro)

TCU - *Transmission Control Unit* (Unidade de Controle de Transmissão)

VAC - Tensão em corrente alternada

VDC - Tensão em corrente contínua

LISTA DE SÍMBOLOS

A - Àmpere, unidade de corrente elétrica

Hz - Hertz, unidade de frequencia

kB - Kilobyte

kb/s - kilobytes por segundo

Mbit/s - Megabits por segundo

Mbps - Megabits por segundo

V - Volt, unidade de tensão elétrica

SUMÁRIO

1. INTRODUÇÃO

Desde que o primeiro veículo à motor foi produzido, no final do século XIX por Carl Benz, o conceito básico de veículos não teve significativas mudanças (HODEL, 2018). Ainda que os componentes e acessórios veiculares tenham sido "assombrosamente" otimizados nesses mais de 100 anos, os veículos atuais, em sua grande maioria, são movidos à combustão (motor elétrico também tem sido implementado há alguns anos), onde transmitem potência propulsora à superfície da estrada através de câmbio, eixo e rodas, que associados aos amortecedores, permitem estabilidade e conforto (WEBER, 2009).

A maior mudança está relacionada aos processos que abrangem o desenvolvimento de veículos. De maneira fundamental, isso se dá pela introdução de sistemas eletrônicos embarcados que têm a capacidade de comandar e controlar as mais diversas funcionalidades do veículo. A integração entre vários subsistemas desse tipo forma o processo de desenvolvimento do veículo.

No domínio automotivo os subsistemas automotivos são gerenciados por unidades de controle eletrônicas denominadas simplesmente por ECU (*electronic control unit* – unidade de controle eletrônico). Um veículo, que é um conjunto de subsistemas, possui diversos tipos de ECUs capazes de gerenciar funções como controle do motor, transmissão, carroceria, entre outros. Para o gerenciamento de cada um destes subsistemas as ECUs são nomeadas de acordo com o propósito de controle (SILVA, 2017).

Nesse âmbito, as áreas de engenharia de software precisam adotar novos processos e métodos que assegurem um nível de segurança e confiabilidade aceitáveis para itens automotivos embarcados, considerando ensaios e modalidades de teste.

Esse trabalho apresenta uma análise do funcionamento de sistemas embarcados, assim como dos meios de certificação e ensaio de testes de software aplicados a esses sistemas, considerando, de maneira geral, o meio automotivo.

1.1. Objetivo Geral

Este trabalho, tem por objetivo realizar um estudo dos modelos de teste existentes e analisar os melhores métodos a serem aplicados à eletrônica embarcada automotiva.

Uma parcela da análise dar-se-á na planta de uma empresa montadora de automóveis do Norte de Santa Catarina (Brasil).

1.2. Objetivos Específicos

Os objetivos específicos do projeto são:

- Estudar os processos de teste e validação utilizados atualmente na empresa em questão para sistemas eletrônicos embarcados;
- Implementar a montagem de uma bancada (simulando um veículo) para aplicação de teste de software, facilitando a manutenção e eventual atualização, assim como uma grande redução de custos nos testes;
- Demonstrar quais são os tipos de testes de softwares/firmwares de sistemas eletrônicos;
- Demonstrar os métodos e procedimentos utilizados nos testes e validação de softwares;
- Identificar as razões pelas quais testes de software devem ser adaptados ao contexto do projeto e às características do produto.

1.3. Justificativa

Considerando que os procedimentos de teste de softwares são muitas vezes subentendidos por quem os realiza, bem como a existência de diversos processos nesse meio levam a constatação de que é necessário fazer uma análise dos processos e da teoria relacionada a este assunto.

Há ainda, o fato de que testes rigorosos de componentes e sistemas, e sua documentação associada, podem ajudar a reduzir o risco de ocorrência de falhas durante a operação. Quando os defeitos são detectados e posteriormente corrigidos, isso contribui para a qualidade dos componentes e sistemas. Além disso, o teste de

software também pode ser necessário para atender aos requisitos contratuais, legais ou aos padrões específicos da indústria.

1.4. Divisão do Trabalho

Já foram vistos neste Capítulo, os objetivos e a justificativa do projeto.

O Capítulo 2 tratará da revisão bibliográfica empregada neste trabalho.

No Capítulo 3 será desenvolvida a análise da certificação e ensaios de testes de software aplicado à sistemas eletrônicos embarcados.

Por último, no Capítulo 4 tem-se a conclusão e algumas sugestões para futuros trabalhos.

2. REVISÃO BIBLIOGRÁFICA

Para a contextualização de como testes de software são aplicados, é imprescindível obter uma base adequada de todo o funcionamento dos sistemas eletrônicos embarcados, desde a definição do hardware até a comunicação entre componentes. Uma definição sobre os tópicos principais de sistemas embarcados não é menos importante.

2.1. Sistemas Embarcados

Existem inúmeras definições sobre o que é um sistema embarcado. Segundo CHASE (2007), um sistema é classificado como embarcado quando este é dedicado a uma única tarefa e interage continuamente com o ambiente a sua volta por meio de sensores e atuadores.

Já segundo o Guia do estudante (2019), sistema embarcado é o nome que se dá a programas e sistemas embutidos em microprocessadores, que executam tarefas específicas em um aparelho.

A alcunha “embarcado” (do inglês *Embedded Systems*) é devida ao fato de que estes sistemas são projetados geralmente para serem independentes de uma fonte de energia. As principais particularidades de categorização de tal sistema são a sua capacidade computacional e a sua independência de operação.

Um veículo de categoria *premium* pode facilmente ser utilizado com um bom exemplo de um sistema embarcado relativamente complexo, onde existem inúmeros sensores, que fornecem informações sobre todo o funcionamento do mesmo. Várias unidades de processamento independentes (ECUs) atuam em diferentes áreas, porém sempre se comunicando, onde é possível captar os sinais de tais sensores e aplicar a ação necessária dependendo do estado de cada um.

2.2. ECU – *Electronic Control Unit*

Até meados do século XX, os veículos eram vistos principalmente como sistemas mecânicos. Mas isso mudou muito com a introdução da eletrônica nos veículos. Em 1970, a unidade de controle eletrônico (ECU) foi introduzida na indústria automotiva.

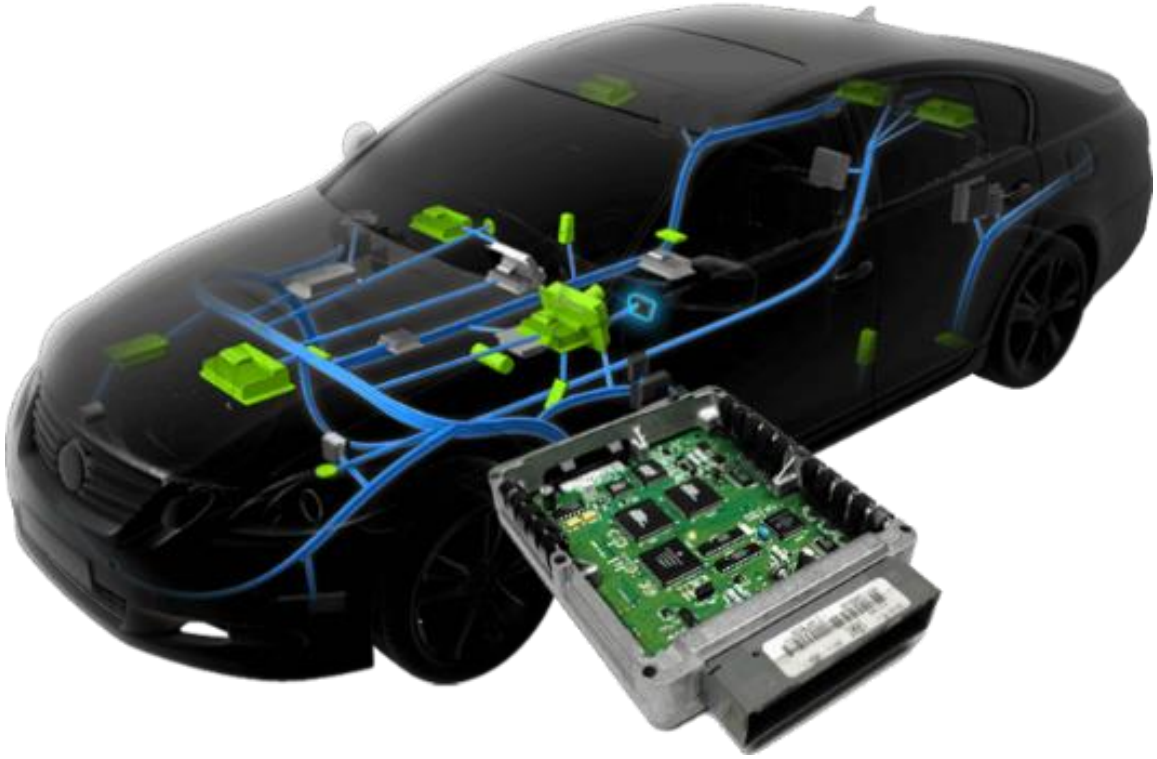
ECU significa unidade de controle eletrônico e é um pequeno dispositivo dentro

de cada veículo que é responsável por controlar uma função específica.

Os veículos modernos podem consistir em mais de 80 ECUs hoje em dia. Eles controlam funções essenciais, como motor ou controle de direção hidráulica e até funções de conforto, tais como abertura de janelas ou ajuste de bancos.

Além disso, as ECUs (conforme exemplo da figura 1) podem controlar a segurança e o acesso de um veículo, como entrada sem chave, *airbags* ou frenagem de emergência.

Figura 1 - ECU Veicular



Fonte:autotechdrive.com. Acesso em 23.07.2021

2.2.1. Funcionamento de uma ECU

O funcionamento de uma ECU é baseado em um dispositivo eletrônico que possui códigos e parâmetros preenchidos em sua memória. Nesse caso há a obtenção de dados de entradas de outras ECUs e sensores. Essas entradas são avaliadas e usadas nos algoritmos para calcular ou determinar a saída. Essas saídas controlam uma função específica.

Um exemplo clássico de como a ECU controla algo é observando como os *airbags* (também conhecido como bolsa de ar ou almofada de ar, é um componente de segurança dos veículos automotores, que pode ser usado também em algumas

máquinas industriais e em robôs de pesquisa, que funciona de forma simples: quando o veículo sofre um grande impacto, vários sensores dispostos em suas partes estratégicas - frontal, traseiro, lateral direito, lateral esquerdo, atrás dos bancos do passageiro e motorista, tipo cortina no forro interno da cabina - são acionados, emitindo sinais para uma unidade de controle, que por sua vez verifica qual sensor foi atingido e assim aciona o airbag que seja mais adequado) são acionados durante um acidente. O carro tem sensores localizados ao seu redor, chamados de sensores de colisão, que informam a ECU quando um acidente ocorre. A ECU então mede a velocidade do veículo quando ele sofre um acidente e, em seguida, usando sua memória de bordo, compara os dados com a especificação e verifica se deve ou não acionar os airbags. Se os dados fornecerem motivos suficientes, a ECU aciona os airbags, tudo isso em meros milissegundos.

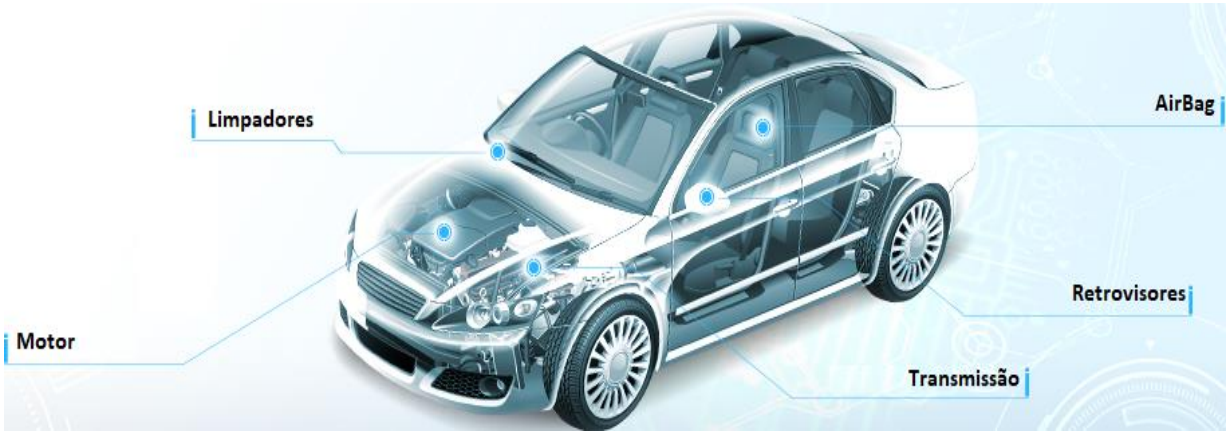
2.2.2. Tipos de ECU

Em veículos com múltiplas ECUs (conforme exemplo da figura 2), elas são divididas nas tarefas que executam. Alguns desses tipos são os seguintes (RANGAM, 2020):

- **Unidade de controle do motor:** Como o nome sugere, ela controla o motor. A Unidade de Controle do Motor é responsável pelo cálculo da carga do motor, entrega de combustível e ignição, resultando em um motor eficiente e altamente funcional;
- **Módulo de controle da carroceria (BCM):** Esta unidade cuida principalmente dos recursos de conforto e segurança do carro. Esta unidade é responsável por sistemas de controle de temperatura, vidros elétricos, travas de portas etc.;
- **Unidade de Controle de Transmissão (TCU):** Esta unidade é principalmente responsável pelas transmissões automáticas. TCU obtém dados de sensores e da unidade de controle do motor;
- **Módulo de controle do trem de força (PCM):** Normalmente, quando as funções da unidade de controle do motor e da unidade de controle da transmissão são combinadas, é denominado como módulo de controle do trem de força;

- **Módulo de controle de telemática:** Normalmente fornece conectividade GPS, Internet e telefone para carros;
- **Sistema de gerenciamento de bateria:** Isso gerencia as baterias recarregáveis em veículos elétricos.

Figura 2 - Exemplos de ECUs



Fonte: silvaco.com. Acesso em 23.07.2021

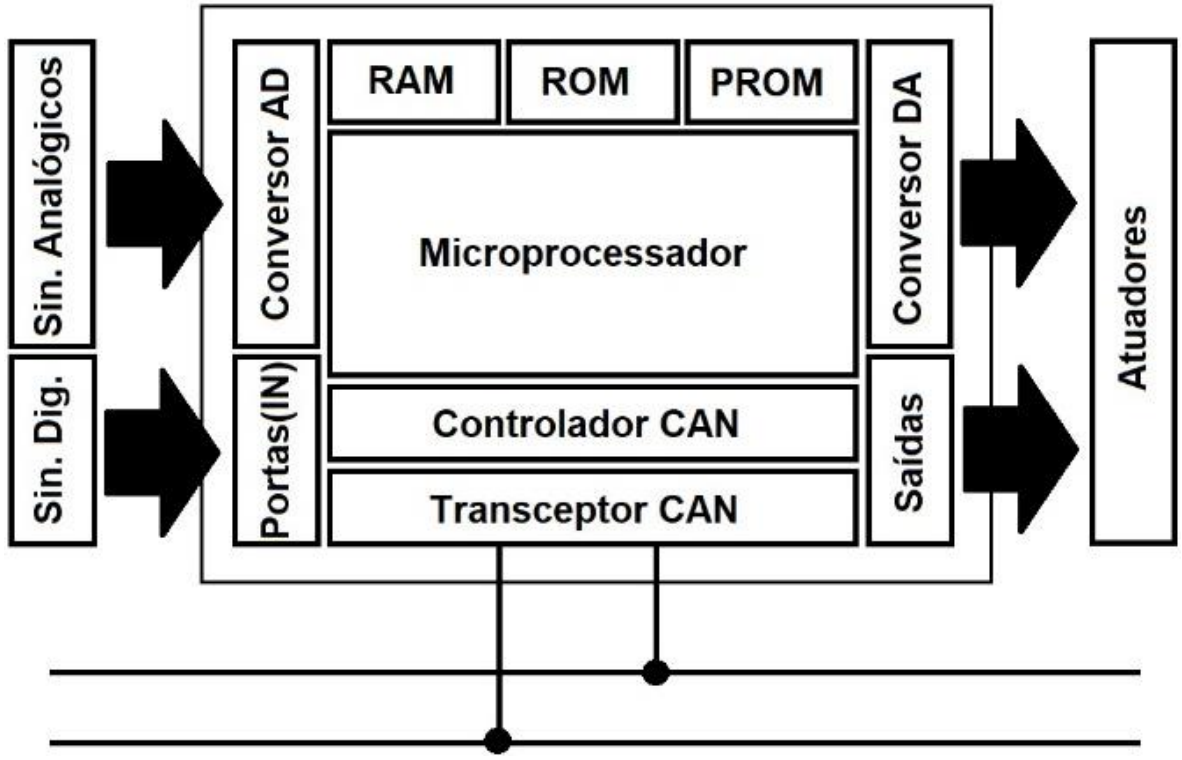
2.2.3. ECU - *Hardware*

De acordo com SUBKE (2019), o hardware de uma ECU consiste em circuitos de entrada (ex.: conversor AD), circuitos de saída (ex.: transistores de potência), uma interface de comunicação de dados (transceptor - um dispositivo que pode transmitir e receber comunicações - CAN) e um microcontrolador. O microcontrolador contém um núcleo microprocessado (CPU) e módulos de memória RAM e ROM.

A interface para comunicação de dados é conectada a uma rede veicular, que interliga várias ECUs.

Os sinais de entrada são gerados por sensores (analógicos ou digitais), tais como os de temperatura (ar, óleo, fluidos) ou por simples chaves on/off. Os sinais se saída são atuadores de acionamento, tais como a bomba de combustível, a unidade de injeção, válvulas do sistema de frenagem, os servomotores dos retrovisores etc.

Figura 3 - Diagrama de Blocos de uma ECU

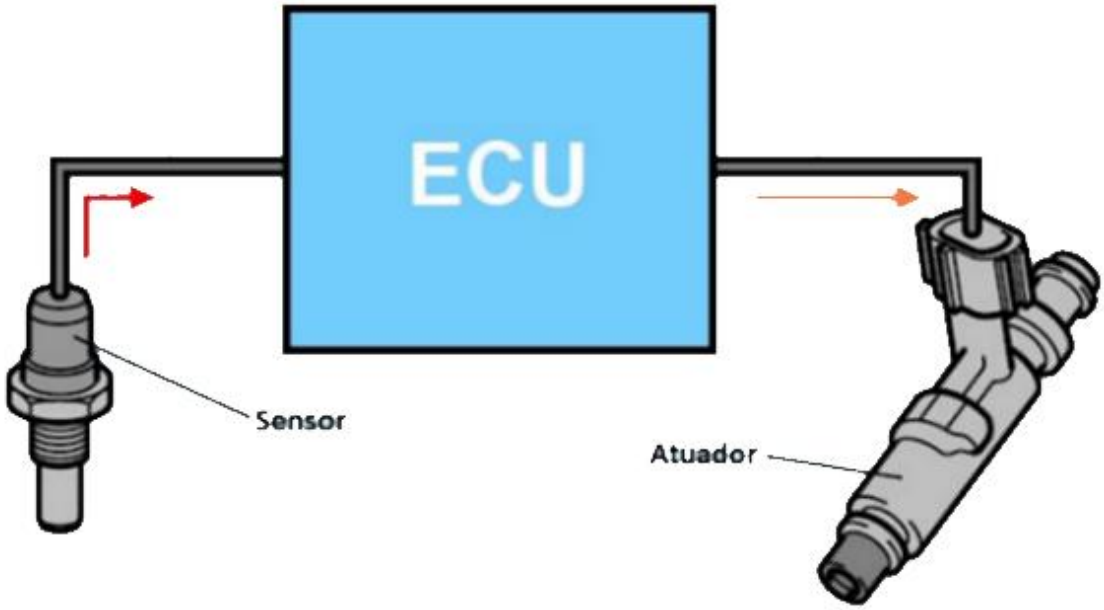


Fonte: adaptado de SUBKE, 2019

2.2.4. ECU - *Software*

ECUs são módulos baseados em microcontrolador, operando de acordo com o modelo IPO (Entrada-Processamento-Saída, do inglês *Input-Processing-Output*) (SUBKE, 2019). Elas convertem sinais de entrada em comandos de saída, realizando dezenas de funções. Esses sinais, obviamente, contêm informações que podem ser “valores físicos”, com uma unidade de medida (temperatura, velocidade etc.), mas também podem ser o estado de uma chave (liga/desliga), que é um sinal binário, sem unidade de medida (posição do acelerador, rotações do motor, pressão do óleo, estado de lâmpadas etc.)

Figura 4 - ECU Operando com o Modelo IPO

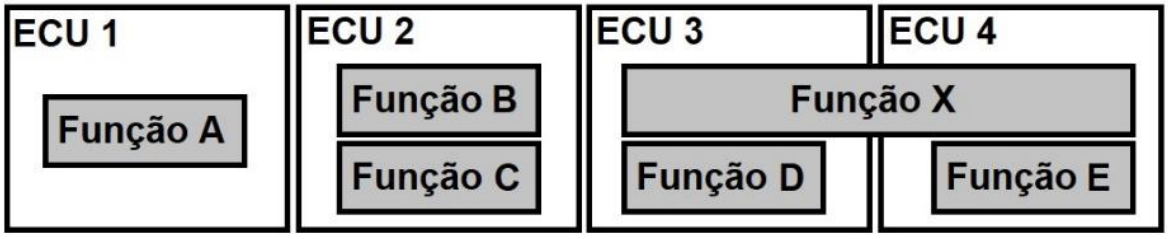


Fonte: docplayer.net. Acesso em 22.07.2021

Um sistema de controle pode ter uma função designada ou várias funções em conjunto (conforme exemplo da figura 5). Adicionalmente, funções podem ser distribuídas entre várias ECUs ou sistemas. Exemplos de funções distribuídas incluem sistemas de segurança, o sistema OBD (*onboard diagnostics* – diagnósticos de bordo) etc.

Devido ao fato de ECUs normalmente não estarem equipadas com um HD (*hard disk* – disco rígido), teclado, *mouse* ou monitor, o desenvolvimento do *firmware* para uma ECU difere fundamentalmente de desenvolvimento de software para Windows, iOS, Android ou sistemas “convencionais”. Em alguns casos, sistemas de operação em tempo real são aplicados, mas se o *hardware* e *firmware* são feitos “sob-medida” para combinar, sistemas embarcados podem funcionar sem nenhum sistema operacional.

Figura 5 - Exemplo de Funcionalidades do Sistema



Fonte: o autor

2.3. Comunicação de Dados

Comunicação é a troca de informação entre, pelo menos, dois parceiros (SUBKE, 2019). Comunicação entre humanos utiliza vários órgãos sensoriais ao mesmo tempo (voz, olfato, contato físico etc.). Comunicação entre humanos e máquinas pode ser através de operações ou programação, tais como girar o volante do carro, apertar um botão ou digitar no teclado. Nesse último exemplo, se o texto segue as diretrizes de programação, isso poderá se tornar um código fonte de um programa (aplicação), que é convertido (compilado) em um código da máquina e finalmente programado na memória de uma unidade de controle (ECU, por exemplo). Atualmente, no âmbito automotivo, *touch screens* (tela sensível ao toque), controle por voz e reconhecimentos de gestos, auxiliam na comunicação do motorista com o veículo.

Máquinas se comunicam com humanos através de alguns sinais, como por exemplo, uma tela que informa o operador sobre o estado atual de tal máquina. Porém até um simples “beep” (sinal sonoro simples) contém informações (ex.: aviso de um e-mail recebido) e um “beep-beep...” contínuo pode, por exemplo, informar à um pedestre que um caminhão está estacionando de ré. Em veículos, um exemplo de retorno/resposta é a pulsação do pedal de freio quando o sistema ABS (*Anti-lock Braking System* - sistema de frenagem que evita que as rodas se bloqueiem, quando o pedal de freio é acionado fortemente, e entrem em derrapagem, deixando o automóvel sem aderência à pista) está ativo ou a simples resposta ao usuário após um clique na tela (conforme exemplo da figura 6).

Figura 6 - Interação Entre Homem e Veículo

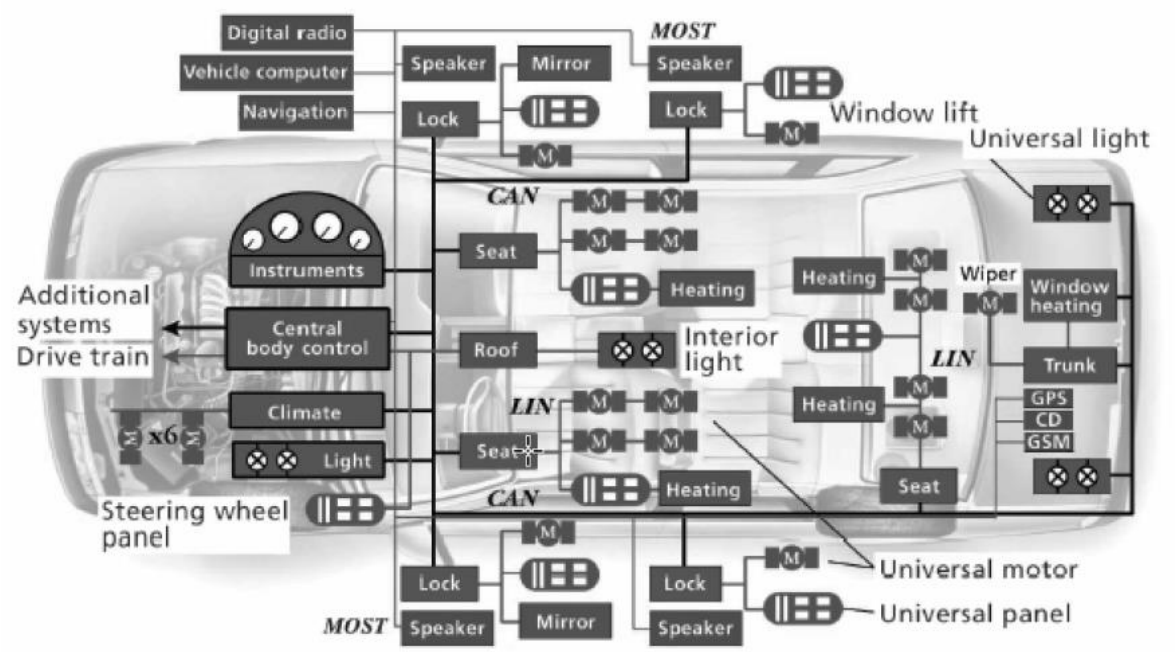
Fonte: motor1.com. Acesso em 24.07.2021

E por último, mas não menos importante, existe a comunicação entre máquinas, que normalmente se dá pela transferência de sinais elétricos através de cabos. Se os sinais forem digitais, a informação está contida em dados. Comunicação de dados é a troca de informação por um código binário. Um sistema de rede de comunicação para controle em tempo real é chamado *Fieldbus* (Segundo NOUVEL et al. (2009), os *fieldbuses* são uma família de redes de comunicação automotiva que evoluíram como uma resposta à demanda para reduzir os custos de cabeamento em sistemas de automação de fábricas. Passando de uma situação em que cada controlador tem seus próprios cabos conectando os sensores para um controlador conectado à ECU que compartilha um barramento. Nesse caso, os custos podem ser reduzidos e a flexibilidade pode ser aumentada)

2.4. Redes Embarcadas

Considerando o grande número de ECUs distribuídas, isso torna o sistema automotivo complexo de várias maneiras. Várias redes diferentes são usadas para atender aos diferentes níveis de requisitos de comunicação (conforme exemplo da figura 7), incluindo segurança crítica e tolerância a falhas. Alta largura de banda se torna necessária para interconectar esses sistemas com previsibilidade. Esta última questão é um grande desafio para os fabricantes de automóveis (NOUVEL et al., 2009). Para reduzir essa complexidade, sem reduzir a segurança, é desejável ter um conjunto limitado de redes. No entanto, não é “em um futuro próximo” que a quantidade de redes pode ser reduzida, podendo atingir apenas um ou dois tipos, já que essa tecnologia forneceria as propriedades de suporte aos sistemas automotivos mais exigentes. Provavelmente o tornaria muito caro. Portanto, é mais provável que algumas tecnologias de rede sejam usadas ao mesmo tempo em que fornecem tolerância a falhas.

Figura 7 - Arquitetura de Rede Veicular



Fonte: Automotive Systems Research Group, 2008

As redes embarcadas aumentaram a funcionalidade e diminuíram a quantidade de cabos. No entanto, o uso de cabos diferentes para as diferentes redes ainda tem a desvantagem de ser pesado, complexo e caro.

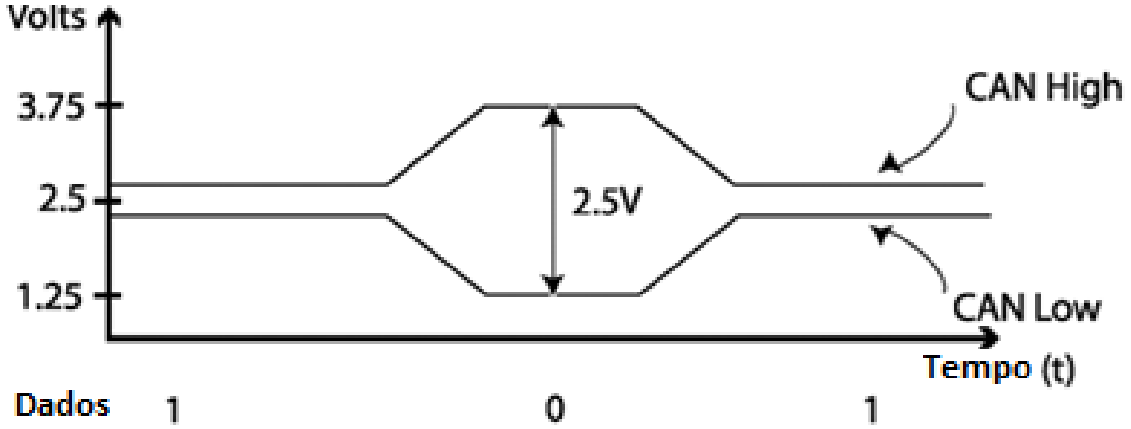
2.4.1. CAN – *Controller Area Network*

CAN foi desenvolvida pelo fornecedor alemão Robert Bosch GmBH como uma rede embarcada para comunicação *on-board* e oficialmente introduzida como uma rede de comunicação serial síncrona no congresso da SAE (*Society of Automotive Engineers*), em Detroit, US em fevereiro de 1986. Um ano mais tarde, os fabricantes de componentes eletrônicos Intel e Philips iniciaram a produção e entrega de um CI (circuito integrado) CAN (SUBKE, 2019). Em 1991, a Bosch introduziu a especificação para CAN 2.0 e em 1993 essas especificações foram padronizadas internacionalmente através da ISO 11898 (NOLTE, 2006).

CAN é um protocolo de comunicação serial que oferece suporte eficiente ao controle distribuído em tempo real com um nível médio de segurança. Na eletrônica automotiva, unidades de controle do motor, sensores, sistemas antiderrapantes etc. são conectados usando CAN com taxas de bits de até 1 Mbit/s.

O barramento CAN usa dois fios dedicados para comunicação. Os fios são chamados de CAN-*high* (alto) e CAN-*low* (baixo). Quando o barramento CAN está em modo ocioso, ambas as linhas carregam 2,5V. Quando os bits de dados estão sendo transmitidos, a linha CAN-*high* vai para 3,75V e a linha CAN-*low* cai para 1,25V, gerando um diferencial de 2,5V entre as linhas, conforme figura 8. Como a comunicação depende de um diferencial de tensão entre as duas linhas de barramento, o barramento CAN não é sensível a picos indutivos, campos elétricos ou outros ruídos. Isso torna o barramento CAN uma escolha confiável para comunicações em rede em equipamentos móveis.

Figura 8 - CAN-low e CAN-high



Fonte: adaptado de axiomatic.com. Acesso em 16.07.2021

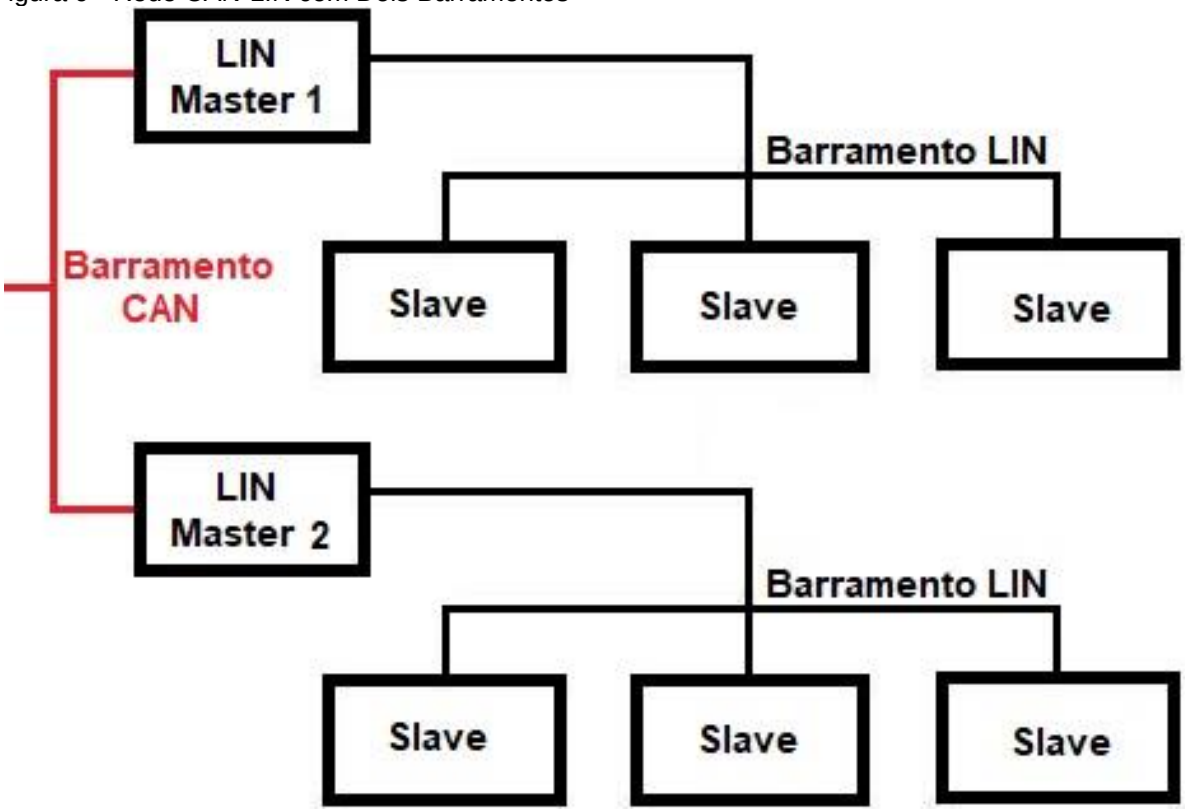
2.4.2. LIN – *Local Interconnect Network*

Idealizado pela Motorola em 1998, o LIN-Bus é um padrão de comunicação veicular utilizada em arquitetura de redes inseridas em carros. A especificação do LIN é mantida pelo LIN-consortium, que é composto por diversos fabricantes de automóveis, como Audi, Volvo, Daimler (Mercedes Benz), Volkswagen e BMW (LIN-consortium, 2003). LIN é um barramento serial, de baixa velocidade (até 20Kbit/s) e baixo custo utilizado para sistemas eletrônicos de controle de carroceria distribuídos em veículos. Ele é comumente usado como um sub-barramento para CAN e/ou *Flexray* (é um protocolo de comunicação para automóveis). Foi desenvolvido pelo consórcio FlexRay Consortium, formado em 2000 pelas empresas BMW, Daimler AG, Motorola e Philips-NXP) e permite uma comunicação eficaz para sensores e atuadores onde largura de banda, velocidade e versatilidade não são necessárias.

De acordo com SUBKE (2019) e considerando as especificações LIN, uma rede com “nós” LIN é chamada LIN *cluster*. Esse cluster consiste em um LIN máster e até 15 escravos (*slave*), que são conectadas via um cabo não isolado.

O LIN máster controla a comunicação no barramento LIN e serve como porta de entrada (*gateway*) para o barramento CAN, conforme mostrado na figura 9. Ele também controla, além da comunicação de dados, a taxa de transmissão de dados. Uma espécie de tabela de agendamento no LIN máster contém a ordem e o intervalo de tempo em que as mensagens serão enviadas (KOPETZ at al., 2002).

Figura 9 - Rede CAN-LIN com Dois Barramentos



Fonte: o Autor

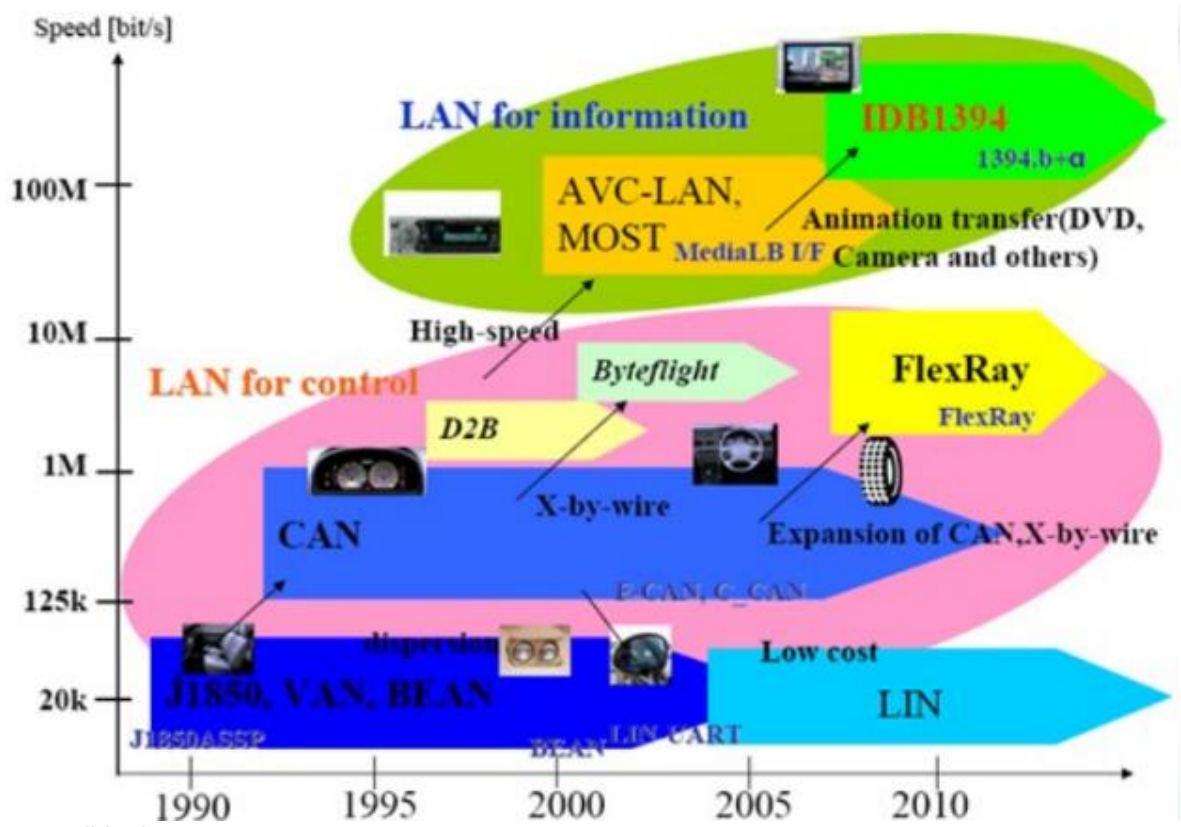
2.4.3. Outros Tipos de Redes Embarcadas

Existem vários outros tipos de rede, no entanto é conveniente mencionar mais dois tipos relevantes, que foram desenvolvidas aplicações multimídia: MOST (MOST, 2006; Nolte, 2006) e IBD 1394 (Navet, Song, Simonot-Lion & Wilvert, 2005; “IBD Forum”, 2008). O MOST, iniciado em 1998, possui velocidade de aproximadamente 150 Mbps. Utiliza fibra óptica plástica como meio de comunicação (e cabos coaxiais) e suporta até 64 dispositivos/ nós (MOST Cooperation, 2008).

Por outro lado, o IDB 1394 oferece suporte a taxa de dados de 100 Mbps e até 63 nós em um cabo de seis vias, com barramento serial, para comunicações de alta velocidade e transferência de dados em tempo real, frequentemente utilizado em áudio digital e vídeo digital. No setor automotivo, o IDB-1394 é uma rede digital que opera a velocidades de até 400 Mbps, transportando vídeo de alta qualidade, som multicanal e dados de alta velocidade para aplicações em veículos. Projetado para aplicações de entretenimento e comunicação, o IDB-1394 permite que os dispositivos operem digitalmente, transportando vários programas de vídeo de alta definição junto com áudio multicanal entre os dispositivos em um veículo (IDB1394, 2008).

Esses dois protocolos relacionam-se no meio automotivo com redes multimídia, armazenamento de dados, aplicativos de navegação e entretenimento. No entanto, apesar de possuírem alta velocidade de transmissão de dados, precisam de fios específicos e permitem apenas comunicações ponto a ponto. Eles atendem especificamente aos requisitos de aplicações multimídia, porém não tem uma utilização muito flexível (NAVET, 2008 E NOLTE, 2006).

Figura 10 - Principais tipos de redes embarcadas

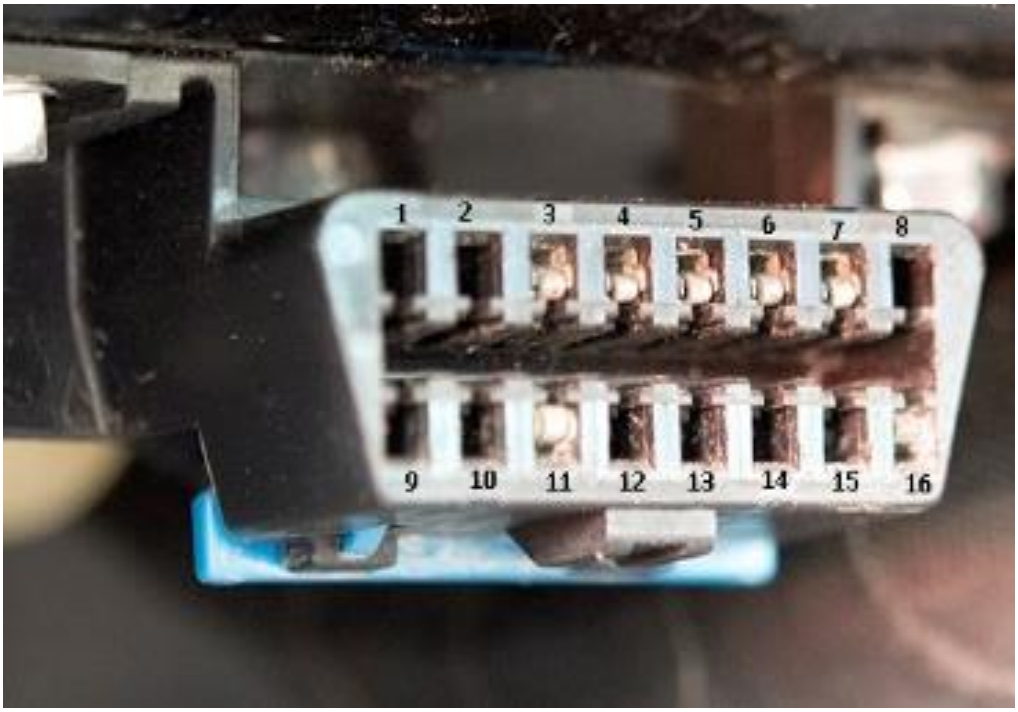


Fonte: slideplayer.com. Acesso em 18.07.2021

2.5. OBD

Segundo a noregon.com, OBD significa *On-Board Diagnostics* e é um sistema de computador dentro de um veículo que rastreia e regula o desempenho de um carro. Este sistema de computador de bordo coleta informações da rede de sensores dentro do veículo, que o sistema pode usar para regular os sistemas do carro ou alertar o usuário sobre problemas. Na figura 11 pode-se visualizar o conector OBD acoplado a um veículo. Esse tipo de conexão é amplamente utilizado pela grande maioria das marcas de veículos no mundo, aplicando desde os modelos populares até nos de categoria premium.

Figura 11 - Conector OBD em um Veículo



Fonte: learn.sparkfun.com. Acesso em 13.11.2021

2.6. Desenvolvimento de Software

O processo de desenvolvimento de software adotado no passado, normalmente, carecia de qualidade e não era confiável. De acordo com NETO (2015), para o mundo corporativo, tais variáveis não poderiam ser ignoradas e o desenvolvimento de software informal não atendia mais. Nesse contexto, foram criadas técnicas para o controle da complexidade dos sistemas, o que culminou com

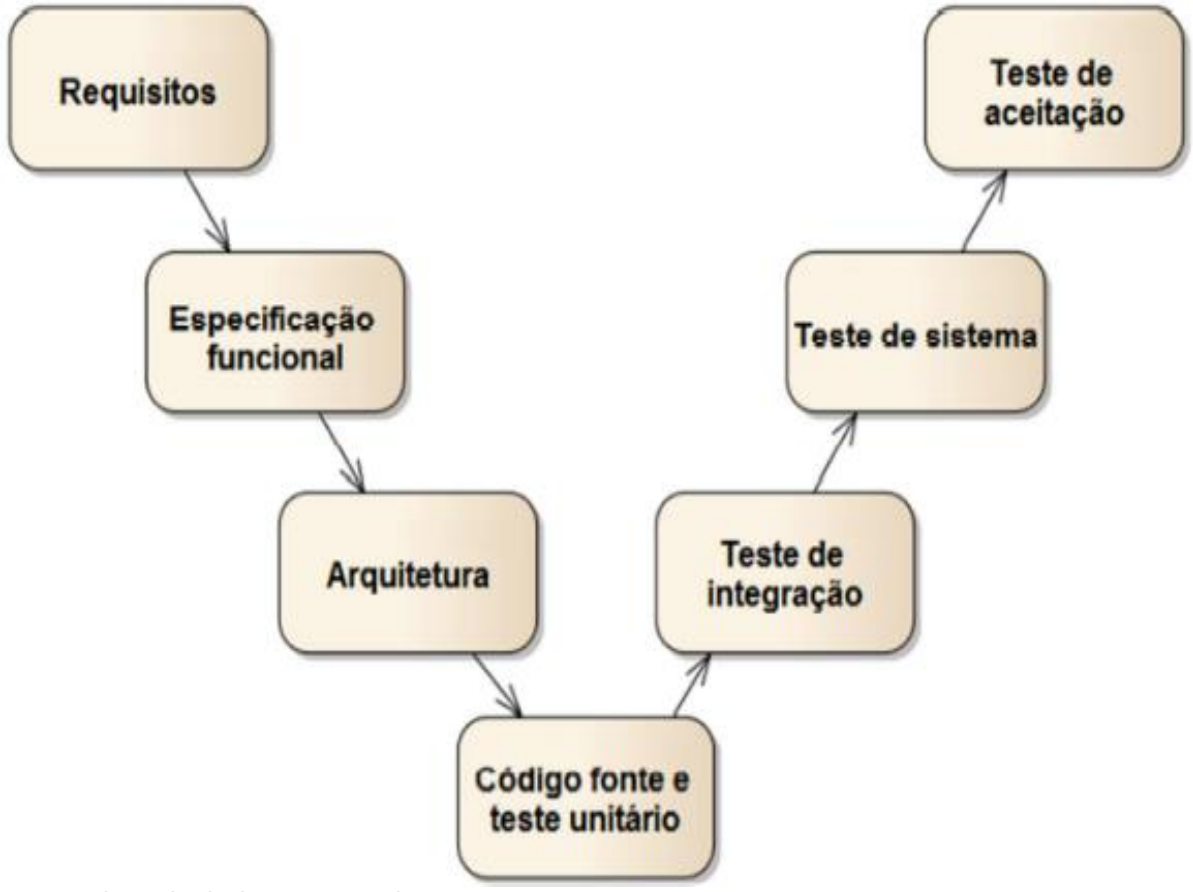
a definição atual de engenharia de software, cujo objetivo era, e continua sendo, desenvolver softwares com alta qualidade dentro de custos e prazo adequados.

Existem diversos modelos de desenvolvimento de software. Ter-se-á a seguir uma abordagem com modelos genéricos.

2.6.1. *V-Model* ou Modelo V

O modelo em V é um conceito utilizado em gestão de projetos que por volta de 1980, com o surgimento da engenharia de software, tornou-se padrão em todos os domínios da indústria de software (ROCHA, 2011). Este é um modelo comumente desenhado com quatro níveis, conforme figura 12. As macro-atividades de desenvolvimentos são sequenciadas e a cada uma é atribuída também uma macro atividade responsável pela garantia da qualidade do produto (NETO, 2015). Essas, por sua vez, representam os níveis de teste que serão tratados mais adiante neste trabalho (ROOK, 1986).

Figura 12 - Modelo V



Fonte: Adaptado de (ROOK, 1986)

O principal conceito do V-model é uma espécie de repetição nos mesmos testes, mas com focos diferentes. Em cada fase do ciclo de vida do desenvolvimento, os testes são realizados levando-se em consideração os níveis de testes. Assim, os testes podem começar desde o início do desenvolvimento, iniciando pelas revisões dos requisitos, inspeções/revisões de código, alcançando até os testes do produto (ROCHA, 2011).

2.6.2. Modelo Cascata

Segundo Wazlawick (2013), o Modelo Cascata (*Waterfall*) começou a ser definido nos anos 1970. Esse modelo considerado o “avô” de todos os ciclos de vida e baseia-se na filosofia BDUF (*Big Design Up Front* – Projetar antes): ela propõe que, antes de produzir linhas de código, deve-se fazer um trabalho detalhado de análise e projeto, de forma que, quando o código for efetivamente produzido, esteja o mais próximo possível dos requisitos do cliente.

O modelo prevê uma atividade de revisão ao final de cada fase para que se avalie se o projeto pode passar à fase seguinte. Se a revisão mostrar que o projeto não está pronto para passar à fase seguinte, deve permanecer na mesma fase (Ellis, 2010).

2.6.3. Demais Processos - Software

Existem muitos processos que podem ser utilizados no desenvolvimento de software, e consequentemente, na verificação dos testes. No entanto, os mesmos não são tão relevantes para a aplicação nesse trabalho e serão apenas listados a seguir (Wazlawick, 2013):

- Codificar e Consertar;
- Sashimi (Cascata Entrelaçado);
- Modelo W;
- Cascata com Subprojetos;
- Cascata com Redução de Risco;
- Modelo Espiral;
- Prototipação Evolucionária;

- Entregas em Estágios;
- Modelo Orientado a Cronograma;
- Entrega Evolucionária;
- Modelos Orientados a Ferramentas etc.

2.7. Os Sete Princípios dos Testes

Alguns princípios relacionados a testes têm sido recomendados nas últimas décadas. Estes servem como diretrizes de uma forma geral para todo o processo de teste e validação (ISQT, 2019):

1. O teste mostra a presença de defeitos, não sua ausência;
2. Testes exaustivos são impossíveis;
3. O teste antecipado economiza tempo e dinheiro;
4. Defeitos se agrupam;
5. Cuidado com o paradoxo dos pesticidas (Se os mesmos testes forem repetidos várias vezes, eventualmente esses testes não encontrarão mais novos defeitos. Para detectar novos defeitos, os testes existentes e os dados de teste podem precisar de alteração e novos testes serem emitidos);
6. O teste depende do contexto;
7. Ausência de erros é uma falácia.

2.8. Testes de Software

Segundo Barbosa (2021), testes de software examinam o comportamento do produto por meio de sua execução, e são formas de se garantir a qualidade. Verificação (no contexto de testes) é o conjunto de atividades que garante que o software implementa corretamente uma função específica, enquanto validação garante que o mesmo corresponde aos requisitos. Erros são cometidos pelos programadores, ocasionando inconsistências, deficiências e comportamentos inesperados (fora da especificação). Falhas são eventos notáveis originados por defeitos.

Pressman (1995) diz-se que as etapas na fase de testes são tão preocupantes como a codificação do sistema, pois pode também acrescentar erros e ao corrigir este

erro na fase de manutenção, o custo é de 60 vezes maior se compararmos a fase do desenvolvimento.

Relacionado à cobertura dos testes tem-se

> Um objeto central de toda a metodologia dos testes é maximizar a sua cobertura, ou seja, a quantidade potencial de defeitos que podem ser detectados, por meio de teste. Deseja-se conseguir detectar a maior quantidade possível de defeitos que não são apanhados pelas revisões, dentro de dados limites de custo e prazos (PAULA, 2010).

2.8.1. Conceitos de Teste de Software

O *Institute of Electrical and Electronics Engineers* (IEEE 610, 1990), exibiu um meio de distinguir os conceitos de falha, erro e defeito:

- **Defeito:** Instrução ou comando incorreto;
- **Erro:** Desvio da especificação;
- **Falha:** Processamento incorreto e comportamento inconsistente.

Conforme menciona Paula (2010, p.461), “um defeito, segundo PMBOK (guia de melhores práticas de gerenciamento de projetos), é uma imperfeição ou deficiência em um componente do projeto na qual esse componente não atende aos seus requisitos ou especificações e precisa ser reparado ou substituído”.

Caetano (2007) diz que para o desenvolvimento de software, os defeitos, apesar do uso de ferramentas e métodos, ocorrem através das pessoas, por isto, testar o software é muito importante, pois é a última etapa antes da entrega do produto ao cliente.

2.9. Níveis de Testes de Software

A seguir, na tabela 1, seguem os níveis existentes para testes de software:

Tabela 1 - Níveis de Teste de Software

Níveis	Meta	Requisitos
Testes de Componentes ou de Unidades	É para garantir que todas as funções e recursos de uma única unidade de código compilável sejam executadas conforme especificação funcional ou estipulada no de design.	Um teste de unidade cobre o teste de uma unidade de software ou um grupo de unidades relacionadas, como uma única entidade. O teste de unidade é realizado isoladamente, usando *test-drives* (teste executado em um veículo, enquanto se dirige o mesmo) ou simulados em bancadas. Essa fase consiste em: • Criação de um plano de teste de unidade • Criação de dados de teste • Realização de testes de acordo com o plano de teste da unidade • Relatórios e revisão dos resultados do teste.
Testes de Integração	É para garantir que todos os subsistemas interagindo em um sistema único façam interface corretamente entre si para produzir os resultados desejados. Além disso, ao tentar atingir esse objetivo, os testes de integração garantirão que a introdução de um ou mais subsistemas no sistema não tenha um efeito adverso na funcionalidade existente.	Um teste de integração cobre o teste de pontos de interface entre subsistemas. O teste de integração é realizado uma vez que o teste de unidade foi concluído para todas as unidades contidas nos subsistemas sendo testados e consiste em: • Criação e plano de teste de integração • Criação de dados de teste • Realização de testes de acordo com o plano de teste de integração • Relatórios e revisão dos resultados do teste.
Testes de Sistema	É para garantir que o sistema funcione de acordo com os requisitos funcionais especificados pelo negócio/usuário.	Cobre o teste de funções dentro do sistema. O teste do sistema é realizado uma vez que o teste do sistema de integração foi concluído e consiste em: • Criação de plano de teste • Criação de dados de teste • Realização de testes de acordo com o plano de teste do sistema • Relatórios e revisão dos resultados do teste Os recursos a serem testados durante o teste do sistema são •Requisitos funcionais • Dependendo do projeto, quaisquer testes de regressão considerados necessários
Testes de Aceitação	É testar com eficácia a qualidade geral de um aplicativo de software por meio da eliminação de defeitos. O objetivo final é fornecer um sistema que possa dar suporte preciso e suficiente tanto para a empresa quanto para seus usuários nas interações do dia a dia.	A etapa final, o teste de aceitação, é conduzido para determinar se o sistema está pronto para o lançamento.

Fonte: ISO/IEC/IEEE 29119

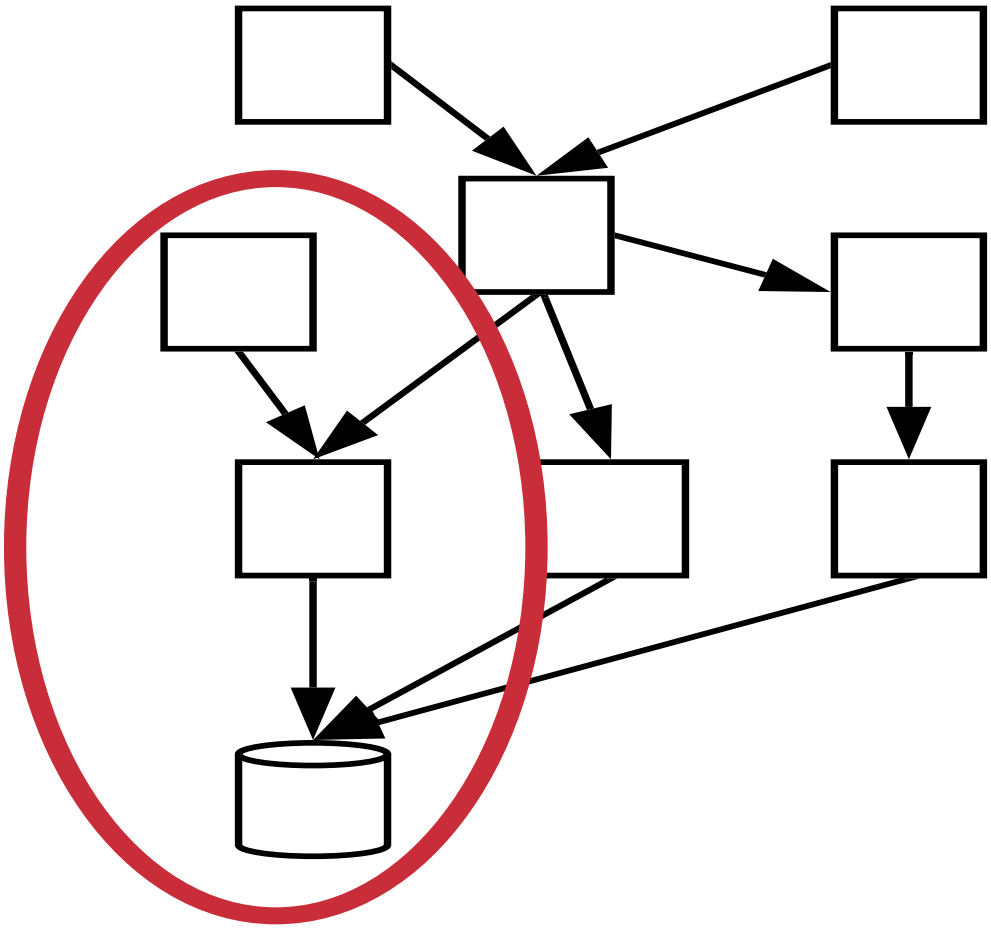
2.9.1. Testes de Integração

Nesse projeto, concentrando-se nas interações dos subsistemas, esse tipo de teste tem os seguintes objetivos:

- Verificar se os comportamentos funcionais e não funcionais das interfaces estão conforme especificação;
- Construir confiança na qualidade das interfaces;
- Encontrar defeitos (que podem estar nas próprias interfaces ou nos componentes ou sistemas);
- Prevenir que os defeitos escapem para níveis de teste mais altos.

Essa fase possui foco total na interação entre dois ou mais componentes (conforme figura 13), não na funcionalidade dos módulos individuais, pois isso deveria ter sido coberto durante o teste de componentes.

Figura 13 - Escopo de testes de integração



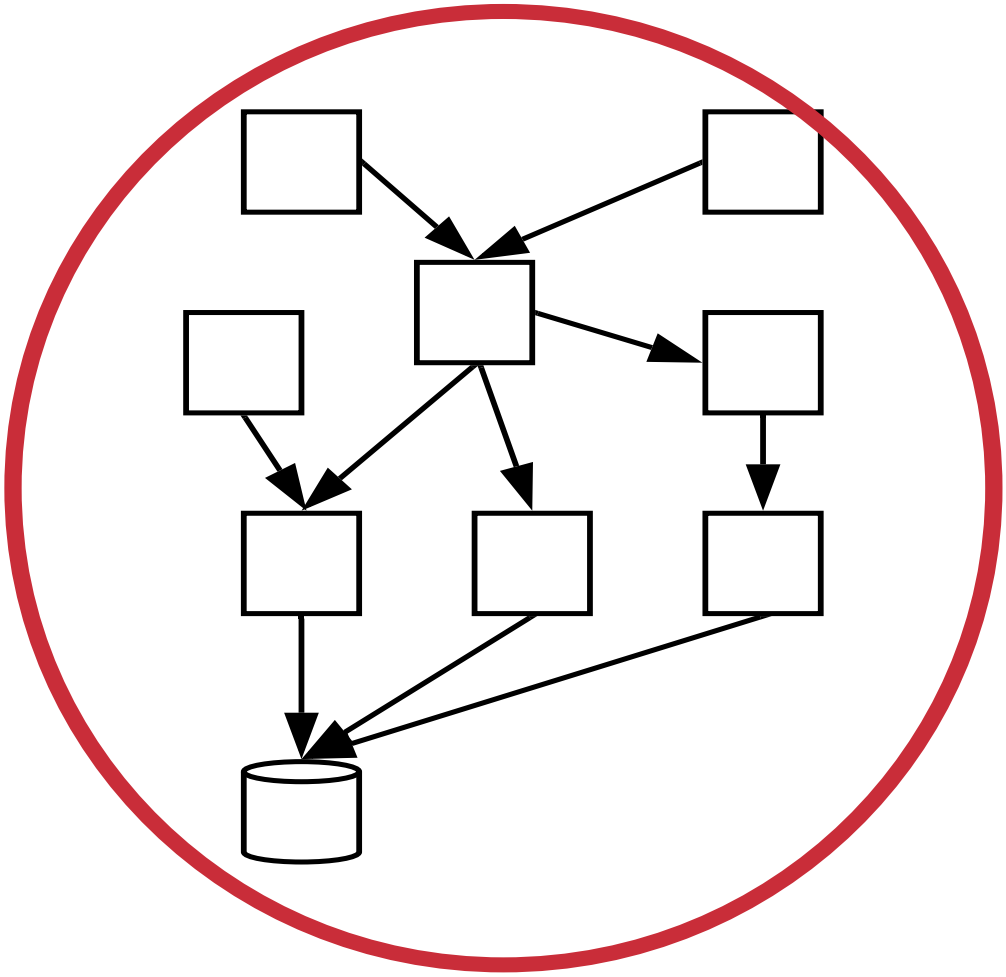
Fonte: VALENTE, 2020

2.9.2. Testes de Sistema

Nesse projeto, concentrando-se no comportamento e nos recursos de um sistema ou produto inteiro (conforme figura 14) e considerando as tarefas de ponta a ponta que o sistema pode realizar e os comportamentos não funcionais que exibe durante a execução dessas tarefas, os objetivos destes testes incluem:

- Verificar se os comportamentos funcionais e não funcionais do sistema são como projetados e especificadas;
- Validar se o sistema está completo e funcionará conforme o esperado;
- Construir confiança na qualidade do sistema como um todo;
- Encontrar defeitos;
- Prevenir que os defeitos escapem para níveis de teste ou produção mais altos.

Figura 14 - Escopo de testes de sistema



Fonte: VALENTE, 2020

2.10. Técnicas de Testes

Para realizar os testes descritos anteriormente há algumas técnicas de testes que devem ser aplicadas. A escolha de cada técnica é estipulada pela área de testes.

A decisão sobre qual técnica usar depende de alguns fatores, entre outros (ISQTB, 2018):

- **Estado da Arte:** A técnica representa o estado da arte atual para esse fim? Padrões como o ISO/IEC/IEEE 29119 auxiliam nesse ponto;
- **Base de teste:** fornece condições de teste adequadas para a técnica? Por exemplo, o testador só pode formar classes de equivalência se a base de teste incluir parâmetros ou variáveis;
- **Teste com base em risco:** significa a identificação dos riscos do produto e a consideração do nível de risco para a seleção das técnicas. Por exemplo, o teste de um valor de limite só faz sentido se houver risco de ocorrência de violações de limite e se o impacto de tais violações constituir um risco.
- **Nível de Teste**: A técnica pode ser razoavelmente usada no nível de teste? Para os testes de caixa preta, o item de teste precisa estar disponível e observável. Por exemplo, o teste de uma classe de equivalência de um sensor no teste do sistema pode ser mais eficiente do que no teste do componente. Se uma técnica de projeto de teste não for utilizável em um nível de teste, o testador deve escolher um nível de teste diferente de acordo com a estratégia de teste.

Para teste de software para indústria automotiva com foco em Infoentretenimento lê-se mais apropriadas as técnicas de *black-box* e regressão.

2.10.1. Testes de *Black-box* (Caixa-preta) ou Testes Funcionais

Segundo ROCHA (2011), o objetivo dessa técnica é exercitar o programa de modo a verificar se ele se comporta de acordo com seus requisitos, ficando o testador alheio à estrutura interna. Para um bom planejamento desse tipo de teste, há a necessidade de que o responsável pelos testes defina combinações de entradas e conheça as saídas esperadas. Em resumo, o sistema é executado com as entradas pré-definidas e as saídas são comparadas com as esperadas.

Para que haja a possibilidade de que um bom teste funcional seja executado, deve-se garantir que os requisitos estejam bem escritos seguindo o princípio de que os requisitos devem ser testáveis.

Conforme Myers (2011), para ter 100% de cobertura dos requisitos é necessário um teste exaustivo com todas as entradas possíveis. Isso torna inviável a cobertura plena. Ao invés disso, o testador deve se basear em técnicas de modelagem que otimizem a busca de áreas nas quais o sistema pode falhar, como partições de equivalência, análise de valor limite etc. Testes funcionais podem ser projetados tanto para encontrar falhas no sistema (nos níveis de sistema ou integração), como para demonstrar que o software implementa os requisitos elicitados (teste de aceitação).

2.10.2. Testes de Regressão

O teste de regressão está profundamente conectado à conservação do software. Alterações nas necessidades dos usuários, avanços e adequações envolvendo condições não funcionais e até mesmo a correção de falhas geram alterações em sistemas na fase de pós-implantação. O teste de regressão deve ser usado para garantir que essas modificações não causem impacto nas funcionalidades já existentes (BELL, 2005).

Graham (2007) define o teste de regressão como o teste repetido após a modificação, de um programa que já foi testado, para descobrir a existência de algum defeito introduzido ou não coberto originalmente como resultado da mudança. Precisa ser realizado quando o software, ou seu ambiente, é alterado.

Myers (2011) defende que não basta apenas executar novamente uma bateria de testes para garantir o teste de regressão. Se for necessário retestar um programa, os casos de teste devem ser reinventados para que mantenham sua qualidade. Para um pacote de testes puramente executado novamente, a possibilidade de descobrir novas falhas é irrisória, o que pode dar a impressão de que a qualidade do sistema está melhor, porém, a verdade é que os testes não exercitaram as áreas do software com as novas falhas.

A reexecução de testes é uma tarefa burocrática e o testador tende a negligenciar a tarefa. Uma forma de diminuir esse impacto é automatizar os testes de regressão mais utilizados (MCCONNELL, 2004).

A quantidade de teste de regressão a ser utilizada no projeto deve ser baseada no risco de não encontrar defeitos no software que estava funcionando previamente (ROCHA, 2011).

2.11. Casos de Teste (*Test Case*)

De acordo com o DATAPREV, caso de Teste é um artefato que estabelece um conjunto de condições e passos para testar um software. Pode ser elaborado para identificar defeitos na estrutura interna do software por meio de situações que exercitem adequadamente todas as estruturas utilizadas na codificação; ou ainda, garantir que os requisitos do software que foi construído sejam plenamente atendidos.

Para a norma ISO/IEC/IEEE 29119-1, trata-se de conjunto de pré-condições do caso de teste, entradas (incluindo ações, quando aplicável) e resultados esperados, desenvolvidos para conduzir a execução de um item de teste para atender aos objetivos do teste, incluindo implementação correta, identificação de erros, verificando a qualidade, e outras informações valiosas.

2.12. Relatório de Falhas, Erros ou Defeitos

De acordo com a norma ISO/IEC/IEEE 29119-2, trata-se da documentação da ocorrência, natureza e status de um incidente. Os relatórios de incidentes também são conhecidos como relatórios de anomalias, relatórios de falhas, relatórios de erros, problemas, relatórios de problemas, entre outros termos.

De acordo com Cristalli (2007), a documentação de um defeito é muito importante para o processo de gestão de defeitos, no entanto, a devida atenção nem sempre é dada, dessa forma, tem-se informações altamente relevantes para a elaboração deste documento:

A. **Evidenciar:** deixar claro a existência do defeito através de arquivos de saída etc.;
B. **Generalizar:** compreender o problema de genericamente, pois o mesmo pode ocorrer em outras situações;
C. **Neutralizar:** descrever os fatos evitando opiniões alheias;
D. **Precisão:** o defeito encontrado deve ser um desvio de comportamento esperado e não falha por entendimento;
E. **Reproduzir:** garantir que o defeito possa ser reproduzido através de etapas;

F. **Revisar:** descrever os passos para reproduzir o defeito, pois a documentação do defeito também é um documento relevante para o projeto;
G. **Resumir:** descrever o defeito de forma sucinta, porém objetiva (CRISTALLI, 2007).

3. DESENVOLVIMENTO

Conhecendo-se os principais protocolos de teste, deve-se partir para o questionamento de porque é necessário testar.

Aplicativos e sistemas de software são uma parte integral da vida cotidiana. Nesse contexto, grande parte dos usuários já teve contato com um software que não funcionava como esperado e teve que lidar com as consequências dessa falha, como por exemplo, a perda de dinheiro, tempo e reputação nos negócios. Assim sendo, o teste e validação é um meio para avaliar a qualidade do software e reduzir o risco de falhas na operação. Existem dois tipos de processos em que os profissionais responsáveis pelos testes podem atuar, o QA (*quality assurance* - garantia de qualidade) e QC (*quality control* - controle de qualidade), definidos como (TARGETRUST, 2015).:

- ***Quality Assurance*:** conjunto de atividades para garantir a qualidade nos processos de desenvolvimento;
- ***Quality Control*:** conjunto de atividades para garantir a qualidade dos produtos/sistemas.

Vale salientar, no entanto, que o processo de validação não consiste apenas em executar testes, mas também planejar, analisar, projetar, implementar, reportar progresso e resultados e, por fim, avaliar o resultado.

Alguns testes envolvem o contato com o componente a ser testado, esses são chamados de testes dinâmicos. Existem também alguns em que não há contato com componente ou sistema, esses são os testes estáticos.

Alguns objetivos típicos da execução de testes são:

- Evitar defeitos avaliando produtos, como requisitos, design e código;
- Verificar se todos os requisitos especificados foram cumpridos;
- Verificar se o objeto de teste está completo e validar se funciona como os usuários e outras partes interessadas esperam;
- Construir confiança no nível de qualidade do objeto de teste;
- Encontrar defeitos e falhas, assim, reduzindo o nível de risco de qualidade inadequada do software etc.

Com relação ao sexto princípio de testes (mencionado anteriormente), onde, “o teste depende do contexto”, pode-se considerar o contexto do ambiente automotivo.

Os fabricantes e fornecedores de automóveis cada vez mais lançam novos modelos de veículos, onde existem objetivos divergentes, complexidade crescente e alta pressão para a inovação. Por outro lado, os padrões e o ciclo de vida dos veículos formam a estrutura que não se altera muito, onde o *tester* (testador - pessoa responsável pela execução de testes) pode atuar. Em resumo, o *tester* está contribuindo com seu trabalho para o lançamento do software e possíveis sistemas.

Para um melhor planejamento e execução, os testes são divididos em níveis (demonstrados no item 2.9), onde cada um é uma parte do processo de teste, consistindo nas atividades realizadas em relação ao software em um determinado nível de desenvolvimento, de unidades ou componentes individuais até sistemas completos.

Esses níveis de teste são baseados no modelo V (apresentado anteriormente), que é considerado um dos métodos mais efetivos para teste e validação no meio automotivo.

Para aplicação dos testes nesses níveis, ainda é possível ramificar cada um deles com técnicas de teste (grupo de atividades destinadas a testar características específicas de um sistema de software, ou uma parte dele), demonstradas no item 2.10.

3.1. Bancada/Rack para Aplicação de Testes

No âmbito automotivo, todos os tipos de testes podem ser utilizados, mas a eficácia depende muito do nível e tipo de aplicação. Com intuito de avaliar a aplicação de certificações e possíveis ensaios de testes de software aplicado a sistemas embarcados automotivos, buscou-se implementar uma rack de teste com todas as principais ECUs para os sistemas eletrônicos, tornando possível a validação de software sem a necessidade de um conjunto veicular completo. Para essa implementação serão analisados os dois tipos considerados mais eficazes para essa aplicação: black-box e regressão.

Para o processo de testagem de software de sistemas eletrônicos embarcados, normalmente é necessário um veículo completo com todas as ECUs, o que onera consideravelmente tal processo. No entanto, ele é de extrema importância para

garantir o nível de qualidade do software. Para amenizar tal questão e servindo como boa alternativa para os testes, é possível utilizar pequenas bancadas (racks), que simulam completamente a eletrônica automotiva com a associação de um grupo de ECUs.

3.1.1. Montagem e Instrumentação da Bancada

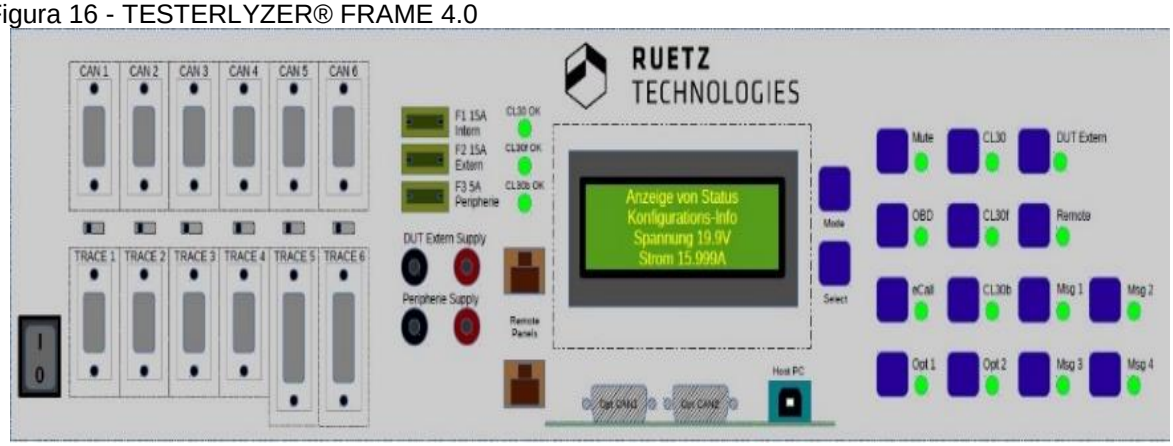
A rack conta com sua estrutura 100% feita com perfis de alumínio 40x40mm. Ela ainda conta com equipamentos para alimentação, sistema de alto-falantes e conectores produzidos e implementados pela empresa Ruetz Technologies (marca registrada), conforme figura 15.

Figura 15 - Estrutura da Bancada de Testes

Fonte: ruetz.de. Acesso em 12.11.2021

A alimentação de entrada é de 100-240 VAC - 50/60 Hz e saída de 13,2 VDC - 15A. O aparelho responsável por essa função (alimentação DC) chama-se TESTERLYZER® FRAME 4.0, demonstrado na figura 16.

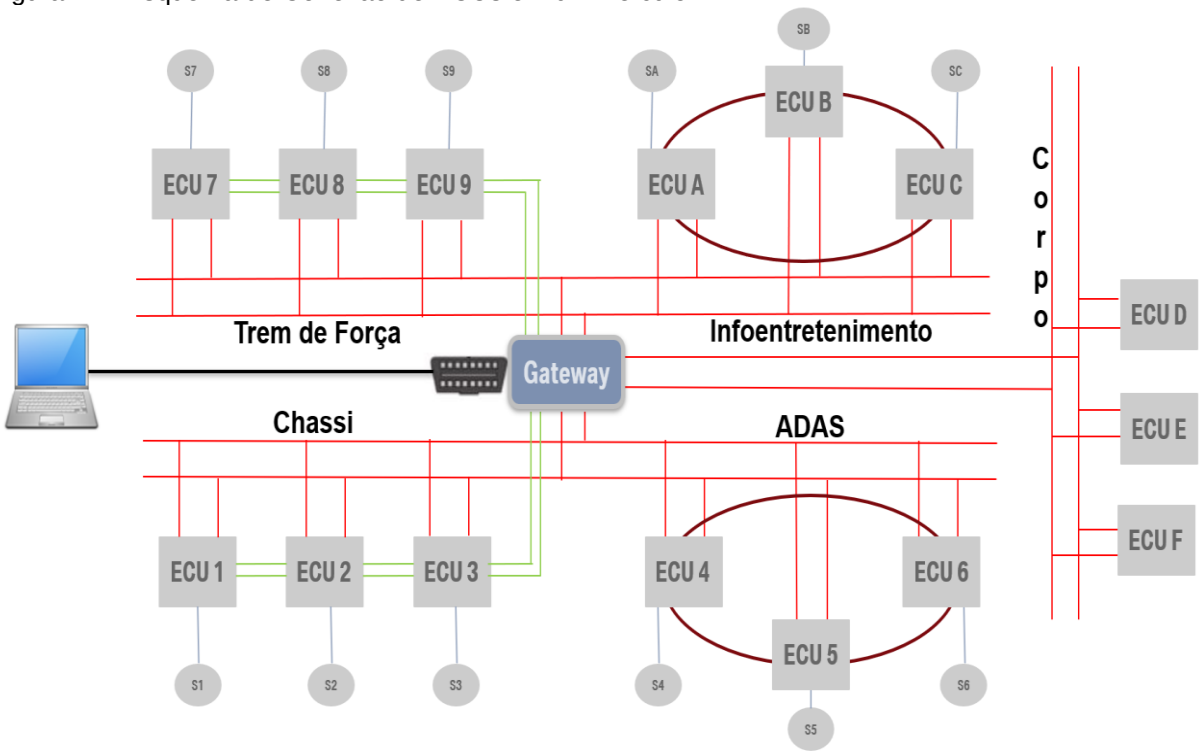
Figura 16 - TESTERLYZER® FRAME 4.0



Fonte: ruetz.de. Acesso em 12.11.2021

Tendo em vista que a alimentação para todas as ECUs já é implementada diretamente à estrutura da bancada, foi necessário entender como cada ECU (ou o grupo delas) seria interligada. Pode-se visualizar o esquema macro na figura 17:

Figura 17 - Esquema de Conexão de ECUs em um Veículo



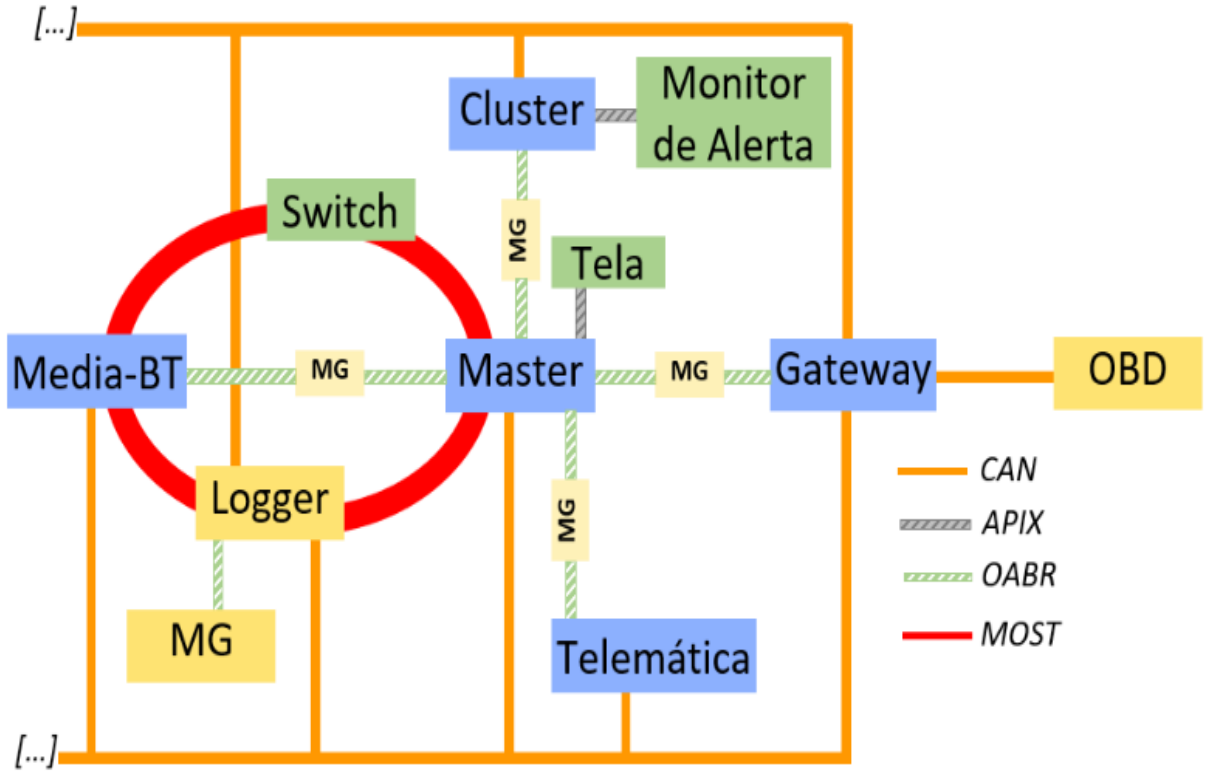
Fonte: o Autor

Onde:

- **Linha Vermelha:** CAN (1Mbps);
- **Linha Cinza:** LIN (20kbps);
- **Linha Marrom:** MOST (22Mbps);
- **Linha Verde:** OABR (100Mbps).

Para a proposta em questão optou-se for focar no subgrupo relacionado a infoentretenimento, do inglês *infotainment* (De acordo com a infopedia.com, sistema multimídia destinado a disponibilizar opções de entretenimento para os ocupantes do veículo, controlar aplicações informáticas, auxiliar a condução etc.), uma vez que é uma área com grande visibilidade e interesse de usuários atuais, necessitando assim um foco maior na validação dele. Na figura 18 é demonstrado o esquema de conexão das redes para ECUs do sistema de *infotainment.*

Figura 18 - Conexões de Redes para ECUs de *Infotainment*



Fonte: o Autor

A instalação iniciou-se com a unidade denominada Master, que é a responsável por controlar todas as outras ECUs relacionadas ao sistema de Infoentretenimento. Ela é interligada às seguintes ECUs por rede OABR:

- **Media-BT:** sistema de entretenimento do assento traseiro;

- **Cluster:** Inclui o velocímetro, conta-giros e hodômetro;
- **Telemática:** responsável por receber e retransmitir os sinais de rede de dados e GPS;
- **Gateway:** “organizador” do tráfego de rede.

Ela ainda possui conexão via rede APIX com a tela de comandos do veículo. Essa tela pode ser identificada como IHM (interface homem-máquina), uma vez que é através dela que há interação entre um usuário e o sistema veicular.

Ainda tratando-se da rede APIX, o cluster é conectado ao monitor de alerta (conforme exemplo da figura 19), que é um refletor que mostra a velocidade atual, o sentido de condução (se uma rota foi traçada via tela) e as informações de multimídia no para-brisas do veículo.

Figura 19 - Exemplo de Monitor de Alerta

Fonte: revistacarro.com.br. Acesso em 05.01.2021

Para todas as conexões OABR entre ECUs há uma interrupção do cabeamento e a conexão com um dos terminais de uma *media gateway* (MG). O MG adquire todas as informações que trafegam nas redes (entre ECUs), converte e armazena em um *logger* (dispositivo com um HD interno).

Todas as ECUs mencionadas, incluindo o *logger*, são também interligadas via rede CAN.

Nota-se na figura 18, que todas as conexões de rede passam pelo ECU *Gateway* (conforme exemplo da figura 20), fazendo o papel de ponte entre todos os tipos de redes e organizando a troca de informações entre ECUs.

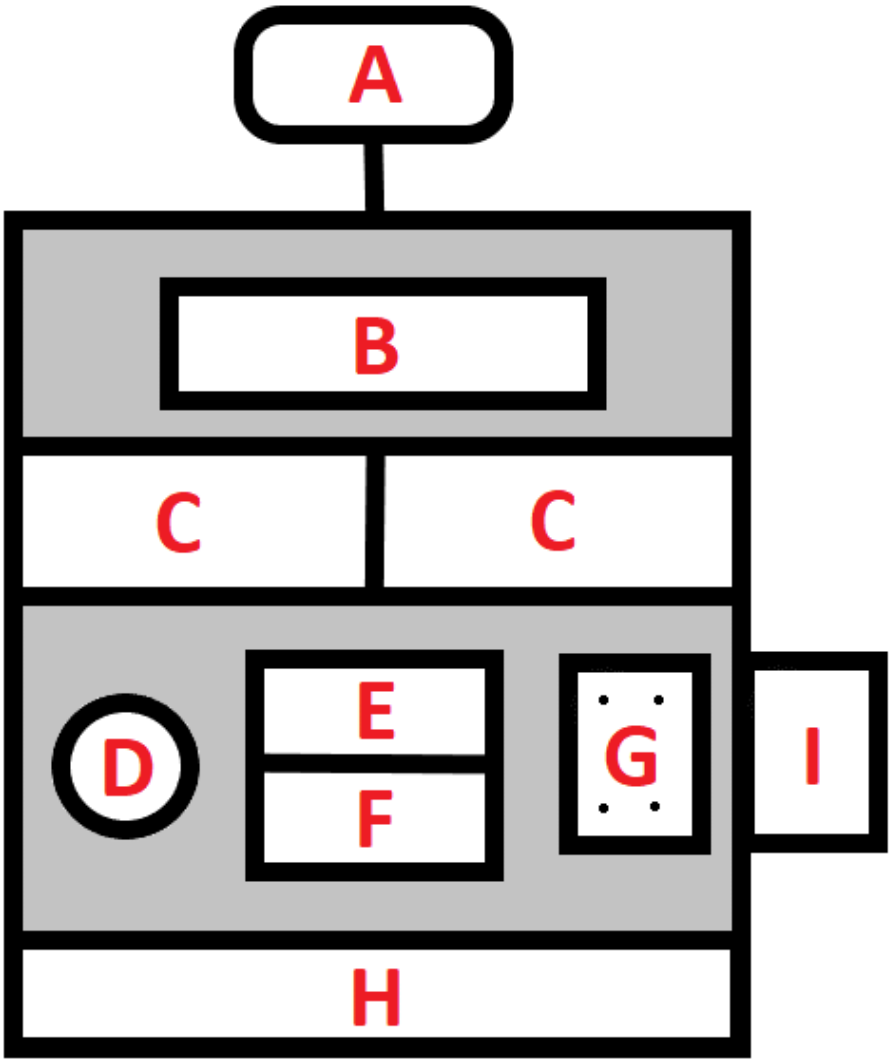
Figura 20 - Exemplo de *Gateway* Utilizado em um Veículo

Fonte: turnermotorsport.com. Acesso em 13.11.2021

Ainda pode-se ver na mesma figura, que o *Gateway* está interligado a uma conexão OBD, que é “a porta” que separa a rede embarcada do mundo exterior. É possível conectar-se ao OBD via cabo, ligar-se a um computador e obter informações de diagnóstico do veículo.

Após a distinção de cada ECU, foi realizada a montagem de ambas na estrutura da bancada de testes, onde a alocação deu-se de acordo com a figura 21:

Figura 21 - Vista Frontal da Bancada



Fonte: o Autor

De acordo com a imagem anterior, as seguintes ECUs ou componentes foram inseridos na parte frontal da bancada:

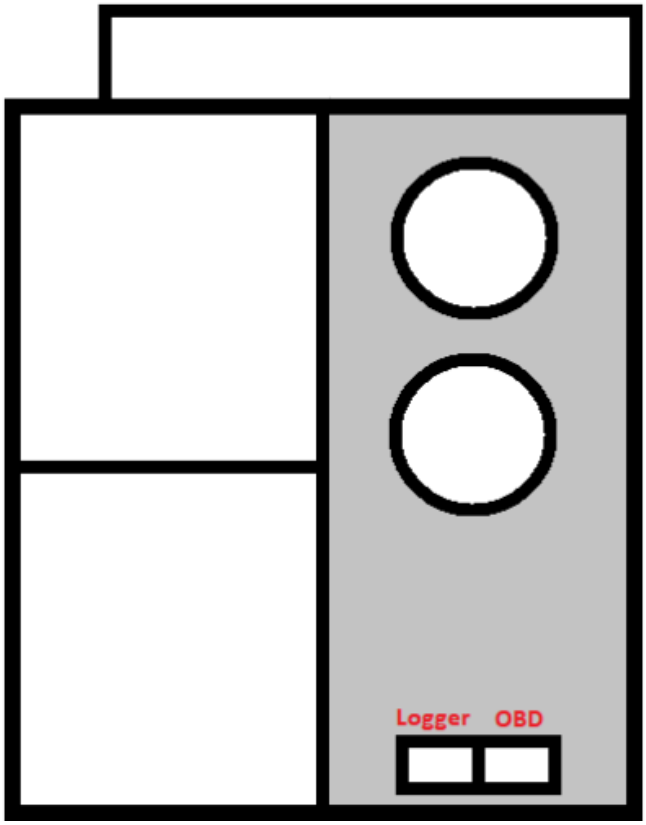
- **A.** Cluster;
- **B.** Tela de controle (painel do veículo);
- **C.** Telas do banco traseiro;
- **D.** Alto-falante principal;
- **E.** ECU Master;
- **F.** ECU Media-BT;
- **G.** Painel com botão de partida, entradas USB e auxiliar;
- **H.** TESTERLYZER® FRAME 4.0;
- **I.** Teclas de controle do sistema.

Partindo para a vista lateral direita (conforme figura 22), tem-se mais dois alto-falantes e as conexões com “o mundo exterior”:

- A conexão OBD mencionada anteriormente foi transformada em RJ 45 para facilitar a conexão (por se tratar de uma bancada);
- A conexão com *logger* também foi feita via cabo de rede (conector RJ 45).

Dessa forma, o usuário pode se conectar ao OBD para fazer a leitura de dados e ao *logger* para baixar informações relevantes (armazenadas no *logger*-HD).

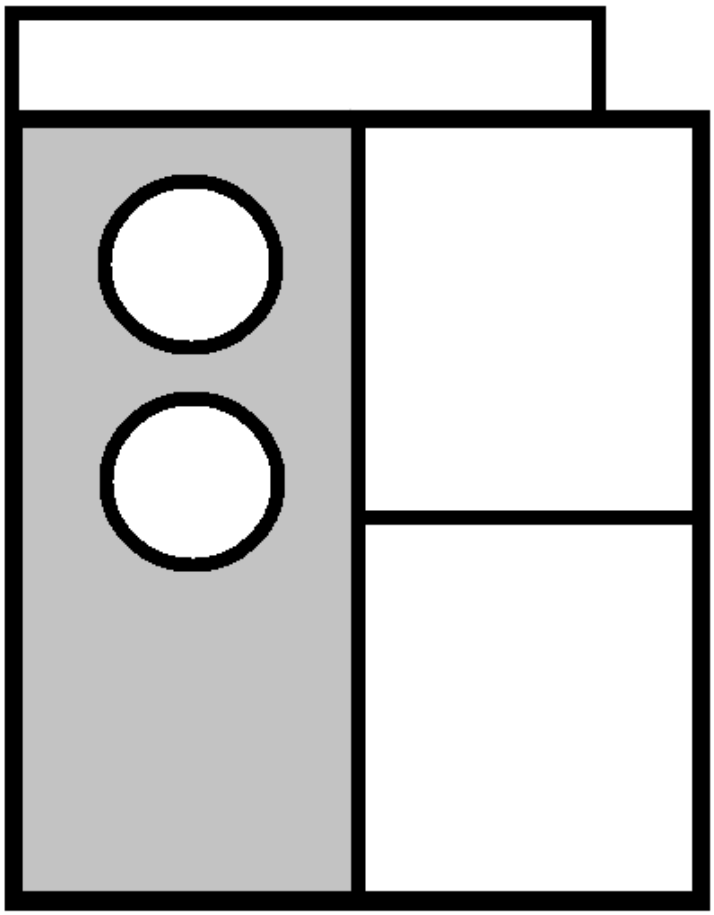
Figura 22 - Vista Lateral Direita da Bancada



Fonte: o Autor

Na vista lateral esquerda (figura 23), tem-se a presença de mais dois alto-falantes e a parte de cabeamento para conexão das ECUs.

Figura 23 - Vista Lateral Esquerda da Bancada



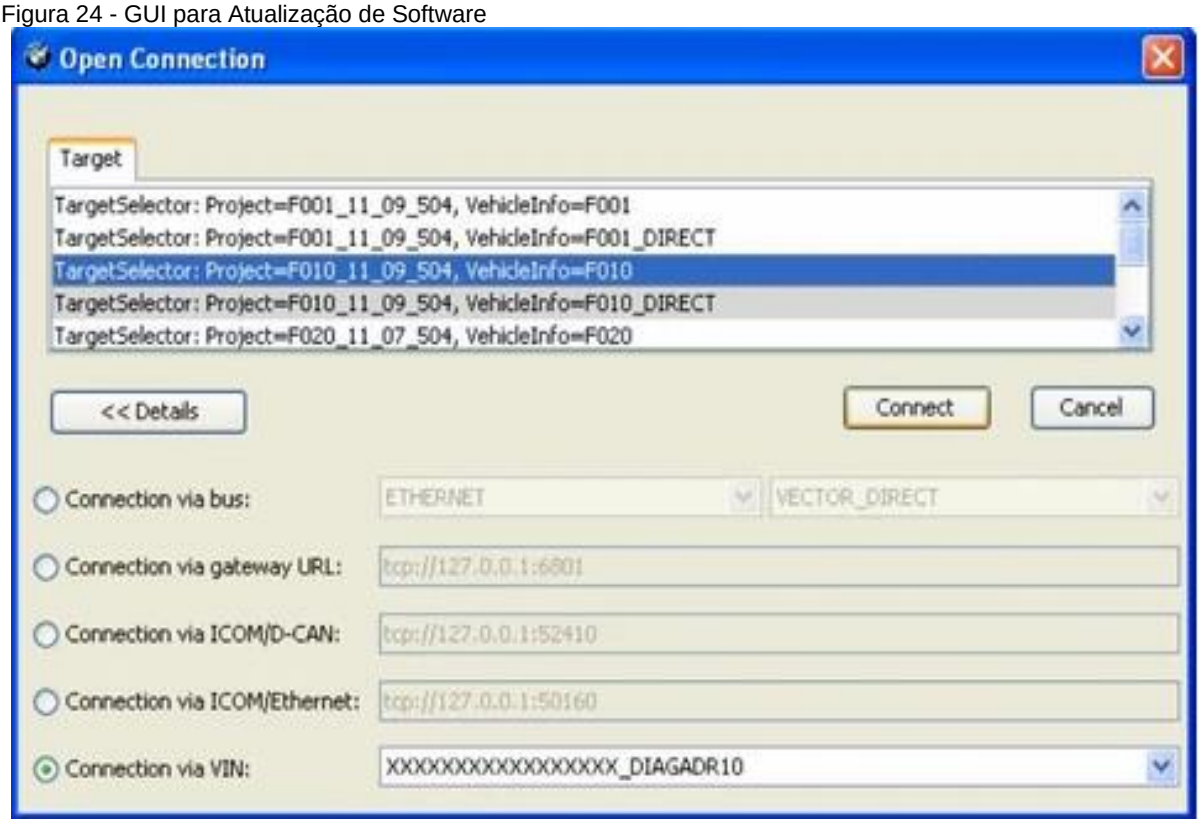
Fonte: o Autor

3.1.2. Atualização ou *Flashing* da Bancada

Após finalizada a montagem e instrumentação da bancada, iniciou-se os protocolos de atualização do software (*flashing*) de todas as ECUs, uma vez que todas dispunham apenas da versão “crua” (de fábrica) e, portanto, não estavam 100% funcionais para o início dos processos de testes de validação.

Iniciou-se pela escolha do software correto de cada ECU, onde estes são disponibilizados em ambiente virtual pela empresa detentora dos direitos de uso da bancada. O carregamento de tais softwares foi feito via software específico (também disponibilizado pela mesma empresa), que por sua vez possui uma interface gráfica (GUI - *graphical user interface*) simples e fácil de utilizar (conforme figura 24).

Figura 24 - GUI para Atualização de Software



Fonte: autosvs.com. Acesso em 23.01.2022

Esse programa (GUI) pode ser instalado em um computador comum e a conexão com a rack também é feita via OBD.

Com a escolha do software correto e após a conexão entre bancada e computador, é possível alterar, adicionar ou remover uma série de parâmetros, tais como:

- Limpar erros registrados no veículo/bancada;
- Desativar avisos sonoros;
- Ativar funções inexistentes até o momento;
- Troca de funções de alguns botões de comando etc.

No entanto, a ideia principal não é alterar parâmetros de fábrica, mas sim atualizar um software obsoleto, que possivelmente contenha correções de erros e melhorias programadas.

Quando já se tem o software inserido no GUI, o último passo antes de iniciar a atualização, trata-se da necessidade de emitir e subir no GUI um arquivo em que consta toda a configuração da bancada/veículo (modelo do veículo, versão de IHM etc.). Esse arquivo contém alguns requisitos desejados e é possível escolher se uma ou mais ECUs serão atualizadas ao mesmo tempo.

Posteriormente, se for necessária a atualização de qualquer software, uma nova versão será necessária, porém em algumas situações, haverá a necessidade de realizar alterações simples em sua configuração, como adicionar ou remover recursos até mesmo o modelo base (modelo de veículo simulado na bancada), trata-se da codificação. Ela pode fazer parte do processo de *flashing*, mas não é exclusiva dele, uma vez que, é possível codificar uma ECU sem a necessidade de *flashing*. As principais vantagens são os ganhos de tempo que são trazidos, e não implica mudanças de software, portanto os riscos envolvidos se o processo de codificação não for bem-sucedido são reduzidos significativamente.

Para que a codificação seja realizada, o arquivo de requisitos precisará ser modificado para que a configuração esteja de acordo com as especificações.

3.1.3. Funcionalidades da Rack

A bancada, após montada e com software em cada ECU, será capaz de disponibilizar todos os recursos de *infotaiment* presentes em um veículo comum, incluindo:

- Comandos de voz;
- Navegação;
- Aplicativos;
- Mídia e streaming;
- Telefonia;
- Serviços remotos;
- Conectividade etc.

Essas funcionalidades têm a mesma efetividade de um veículo completo, porém com algumas pequenas limitações, tais como: ativar a navegação para um destino específico e dirigir até o mesmo. Por motivos óbvios, não é possível “dirigir uma bancada”, mas a navegação ainda pode ser adicionada no sistema. Isso seria possível com a automatização de sistemas, porém essa questão será tratada no tópico de trabalhos futuros.

3.2. Padrões de Testes de Software

Após as montagem e instrumentação da bancada, é possível iniciar a implementação dos testes. Eles foram baseados na ISO/IEC/IEEE 29119, que é um conjunto de padrões internacionalmente aceitos para teste de software que pode ser usado em qualquer organização ou ciclo de vida de desenvolvimento de software. Esses padrões são únicos e internacionalmente reconhecidos e aceitos para teste de software.

Existem atualmente cinco padrões no conjunto citado:

- **ISO/IEC/IEEE 29119-1:** Conceitos e Definições;
- **ISO/IEC/IEEE 29119-2:** Processos de Teste;
- **ISO/IEC/IEEE 29119-3:** Documentação de Teste;
- **ISO/IEC/IEEE 29119-4:** Técnicas de Teste;
- **ISO/IEC/IEEE 29119-5:** Teste baseado em palavras-chave.

Alguns padrões relacionados podem ser incluídos:

- **ISO/IEC 33063:** Modelo de Avaliação de Processo;
- **ISO/IEC 20246:** Avaliações de produtos.
- **NBR/ISO/IEC 12207:** Tecnologia de informação – Processos de ciclo de vida de software

3.3. Implementação dos Testes

Com a cobertura da norma ISO/IEC/IEEE 29119, propõem-se a aplicação de testes à bancada de acordo com os níveis estipulados anteriormente. Todas as fases (demonstradas no item 2.9), exceto a última, consistiram em 3 processos de gerenciamento de teste (ISO/IEC/IEEE 29119, 2013):

- Planejamento de teste;
- Monitoramento e controle de testes; e
- Conclusão do teste.

Nesse caso, tratando-se da parte de planejamento, programou-se o recurso que foi utilizado para execução de testes (nesse caso a bancada), assim como também a criação de casos de teste, de acordo com a funcionalidades que se deseja testar. Os

casos de teste são os documentos principais para planejar, executar e reportar cada teste específico.

Na questão do monitoramento e controle de testes, foi necessário a execução do mesmo caso de teste várias vezes, uma vez que de acordo com as regras de probabilidade e estatística, quanto maior a amostragem, maior será a confiabilidade. Isso torna-se evidente, à medida que se um teste for executado uma vez e houver falhas, não faz sentido reprovar toda uma versão de software. Mesmo caso se um teste for executado apenas duas vezes e houver uma falha, pois a taxa de rejeição seria de 50%. Taxa essa que rejeitaria qualquer software testado.

Os resultados obtidos serão vistos na sequência desse trabalho. Essa etapa é relacionada à conclusão de testes e será findada se o resultado for positivo, relacionado aos critérios. Caso haja resultados negativos, ou seja, falhas, a última etapa para conclusão é o relatório de falhas.

Considerando os tipos de teste mencionados anteriormente, os testes de componente não foram considerados a fundo nesse projeto, uma vez que todas as unidades já foram previamente desenvolvidas e testadas por empresas qualificadas: os componentes para montagem da bancada foram recebidos com certificado de validação e qualidade. Os testes de aceitação (última etapa), também não serão aplicados à bancada, uma vez que nesse caso, alguns clientes reais serão pré-selecionados e instruídos a utilizar o software em questão, validando o mesmo.

3.4. Criação de Caso de Teste (*Test Case*)

O caso de teste é o documento que guia o testador, indicando as entradas e saídas de cada funcionalidade, de acordo com o sistema IPO (mencionado anteriormente). Um modelo de caso de teste foi proposto, em que o intuito é abranger da melhor maneira possível a validação do software de sistemas embarcados automotivos. A seguir, tem-se o modelo proposto (baseado na norma ISO/IEC/IEEE 29119-3 – Item 3):

- **Título:** Nome do caso de teste com informações claras e específicas;
- **Precondições:** Consta como a adição das condições iniciais necessárias para execução do teste (ex. celular conectado ao veículo). Aqui existe apenas uma descrição e nenhum resultado ainda é esperado;

- **Passos de Execução:** Consta como a adição dos passos de setup e as ações a serem executadas pelo testador (ex. clique no botão de login). Podem ser adicionados quantos passos forem necessários;
- **Passos de Validação:** Consta como a verificação dos os resultados após cada passo de execução (ex. Login efetuado). O número dos passos de validação deve ser exatamente o mesmo dos passos de execução.

Cada caso de teste foi elaborado de forma individual para cada funcionalidade, para que todas pudessem ser validadas. Caso houvesse a inclusão de mais funcionalidades em um mesmo caso de teste, ambas poderiam ser reprovadas na validação em caso de resultado negativo. Casos de teste individuais permitem que o testador isole falhas e não comprometa toda a funcionalidade do sistema.

Para a criação de casos de teste adotou-se uma linguagem simples e clara, porém que continha todas as informações relevantes. Deve ser possível que qualquer pessoa entenda o caso de teste, mesmo que nunca tenha tido contato com o sistema testado.

Para a validação de testes de sistema e integração, os seguintes casos de teste foram criados:

- **TC01** - Conexão via Apple Carplay;
- **TC02** - Envio de Serviço Remoto via Smartphone;
- **TC03** - Verificação das informações de Trânsito em;
- **TC04** - Chamada de Emergência;
- **TC05** - Funções Inteligentes - Regras de Climatização;
- **TC06** - Chamadas Telefônicas em Paralelo;
- **TC07** - Navegação de Aprendizagem;
- **TC08** - Trava de Velocidade - App de Notícias;
- **TC09** - Comandos de Voz - Ventilação;
- **TC10** - Sintonia de Estações de Rádio.

A descrição completa deles está descrita no apêndice A.

Casos de teste, normalmente, já são disponibilizados anteriormente ao testador, seja por um time de desenvolvedores, por uma empresa ou até mesmo pelo responsável pela validação de sistemas de software. A criação dos mesmos nesse projeto deveu-se à necessidade de cobrir e demonstrar todo o processo de validação.

3.5. Aplicação das Técnicas aos Níveis de Teste

Existem situações em que a bancada de teste (ou o próprio veículo) precisa ter o software, de uma ou mais ECUs, atualizado em determinados períodos ou até mesmo quando houver um erro muito grave reportado. Para esses casos, haverá a implementação de um novo código pelos desenvolvedores do software. Essa alteração pode fazer com que novos erros apareçam e até mesmo erros de versões (de software) passadas que “voltam à vida”. Nota-se que para cada alteração de software uma nova validação é necessária e para a bancada de testes o mesmo protocolo foi aplicado. Os testes foram aplicados após atualizações de software, então a técnica de testes de regressão foi aplicada.

Pode-se afirmar também que os testes foram executados sem a interação com o código, “injetando” um valor de entrada e esperando um resultado na saída (novamente o sistema IPO). Tal técnica utilizada foi a de testes de caixa preta, onde testes exaustivos foram aplicados, mas com a mesma interação, com o equipamento, de um usuário de veículo comum.

Vê-se que duas técnicas de teste foram utilizadas simultaneamente (regressão e caixa preta), buscando trazer a qualidade almejada ao software e ao mesmo tempo aproximando-se da utilização de um usuário comum.

Tratando-se dos níveis de teste anteriormente mencionados, focou-se nas fases de integração e sistema, abrangendo o maior número de ECUs possíveis (na tabela 2):

Tabela 2 - ECUs Testadas por Níveis

Caso de Teste	Nível de Teste	ECUs Testadas
TC01	Integração	Master, telemática, tela
TC02	Integração	Master, telemática, tela
TC03	Integração	Master, telemática, cluster, tela
TC04	Integração	Master, telemática, tela
TC05	Integração	Master, telemática, Media-BT, tela
TC06	Integração	Master, telemática, tela
TC07	Sistema	Todas
TC08	Integração	Master, telemática, tela
TC09	Sistema	Todas
TC10	Integração	Master, telemática, tela

Fonte: o Autor

Nota-se que há um número bem maior de testes de integração em relação aos testes de sistema, uma vez que dependendo da especificidade de cada função, torna-se difícil englobar todas as ECUs em um único teste.

Com a relação às ECUs validadas em cada caso de teste, é possível evidenciar que a ECU master aparece em todos eles, uma vez que ela é a principal unidade do sistema e responsável por todo o processamento de informações e dados. Mesmo caso para a telemática, que recebe todos os sinais de rádio, GPS e dados e para a tela, que é a principal interface com o usuário.

Por fim, é importante mencionar que as informações dos testes realizados e falhas encontradas estão todas armazenadas no *logger*.

3.6. Resultado dos Testes

Para a validação da qualidade do software instalado na bancada de teste, buscou-se utilizar todos os dez casos de teste mencionados anteriormente, com os seguintes resultados obtidos (na tabela 3):

Tabela 3 - Resultado da Aplicação de Casos de Teste

Caso de Teste	Número de Execuções	Resultados Positivos		Falhas	
		Total	%	Total	%
TC01	20	18	90%	2	10%
TC02	20	17	85%	3	15%
TC03	20	19	95%	1	5%
TC04	20	20	100%	0	0%
TC05	20	20	100%	0	0%
TC06	20	19	95%	1	5%
TC07	20	17	85%	3	15%
TC08	20	20	100%	0	0%
TC09	20	19	95%	1	5%
TC10	20	20	100%	0	0%

Fonte: o Autor

Nota-se que para alguns casos de teste, o percentual de falha está entre 5 e 15%, considerando que cada teste foi executado 20 vezes. Tais falhas encontradas são as seguintes:

- **TC01:** Falha na conexão do Apple Carplay (2x);
- **TC02:** Falha no recebimento de endereço (3x);
- **TC03:** Informações de trânsito divergem do real (1x);
- **TC06:** Chamada não foi recebida (1x);
- **TC07:** Navegação não foi aprendida (3x);
- **TC09:** Comando de voz não identificado (1x).

Essas falhas foram encontradas através da análise dos critérios de entrada e saída (IPO) definidos em cada caso de teste. Através dos critérios de saída, foi possível definir quais condições deveriam ser alcançadas para declarar um nível de teste ou um conjunto de testes concluídos.

Os critérios de entrada e saída foram definidos para cada nível de teste e tipo de teste e são diferentes, baseados nos objetivos do teste.

Para cada falha encontrada, independente se uma ou mais vezes, um relatório de defeito precisou ser emitido. O software em questão precisou continuar em sua fase de desenvolvimento e todo o processo se repetirá até que atinja uma qualidade adequada.

3.7. Relatório de Falha

Conforme mencionado, um relatório foi criado para cada falha encontrada, com o intuito de informar os responsáveis sobre tais problemas e tentar antecipar a revisão do software testado e o recebimento de uma nova versão. Para cada relatório, os seguintes passos foram seguidos:

- **Dados da Bancada:** adicionou-se as versões de software de cada ECU testada, uma vez que para testes de sistema ou integração, dados de todas as ECUs verificadas precisarão ser analisados;
- **Registro:** nesse passou o exato momento em que a falha ocorreu foi registrado, considerando data e horário;
- **Pré-condições:** inclui todas as condições pré-estabelecidas antes do erro ocorrer. No caso todas as condições descritas no caso de teste;
- **Passo para Reproduzir o Erro:** inclui-se, especificamente, todos os passos seguidos pelo testador e que são necessários para reproduzir a falha;

- **Resultado Esperado:** Trata-se do que se espera de resultado no caso de testes, considerando os valores de saída (sistema IPO);
- **Resultado Obtido:** Tratou-se dos valores de saída, porém que diferem dos resultados esperados;

Além de todos os passos anteriores terem sido seguidos, evidências também foram adicionadas, com a intenção de comprovar as falhas. Considerou-se vídeos e fotos para as evidências, assim como a inclusão dos dados das ECUs, obtidos através do *logger* da bancada.

4. CONSIDERAÇÕES FINAIS

4.1. Conclusão

Foi apresentada a análise de protocolos de teste de software para sistemas embarcados automotivos. Testes esses baseados em uma bancada de testes (simulando o funcionamento do sistema de Infoentretenimento) que foi montada e codificada para tal uso. A utilização da mesma possibilitou a facilidade de manutenção e a redução considerável de custo em relação à um veículo completo. A pesquisa identificou práticas e padrões que podem ser adotados de forma a gerenciar corretamente o processo de teste de software, modelar e executar os casos de teste, reportar os incidentes encontrados e, por fim, medir a qualidade do trabalho realizado.

A ideia do trabalho é mostrar que a área de teste não engloba um ato de complicar o mundo do desenvolvimento de software, mas sim uma facilitar, onde após encontrar erros, há a necessidade de correções, e, consequentemente se nota uma melhora nos produtos, na geração de conhecimento e minimização das insatisfações futuras.

Deste modo, há a necessidade de compreender melhor como as metodologias podem influenciar na evolução e eficiência da entrega do produto e que com o auxílio de outras vertentes metodológicas pode-se alavancar ainda mais os processos inovando a maneira das possíveis formas de trabalhar com as metodologias.

Para tudo isso se manter, as normatizações impostas pelas ISOs (e demais organizações) verificam se a qualidade do software e do processo estão sendo empregadas devidamente. Além de mostrar as melhores maneiras de empregá-las.

4.2. Oportunidades de Melhoria

À medida que o processo de teste for amadurecendo, sugere-se, como trabalhos futuros, a inclusão de metodologias ágeis no processo de teste e validação para tornar o processo mais rápido e efetivo.

Além disso, sugere-se a automação gradativa dos testes funcionais resultantes desse trabalho, como parte do conjunto de testes de regressão, de forma a executá-los na integração contínua dos sistemas. Os testes de regressão são sempre fortes

candidatos à automação, pois a execução repetitiva de casos de testes costuma transformar-se em negligência (mesmo que involuntária) por parte do testador.

REFERÊNCIAS

Automotive Systems Research Group, 2008. Disponível em: http://www.ttcan.com/ASRG. Acesso em: 30.06.2021.

CAETANO, C., **Automação e Gerenciamento de Testes: Aumentando a Produtividade com as Principais Soluções Open-Source e Gratuitas**, e-book disponível <http://www.linhadecodigo.com.br/ ebook.aspx?id=2951> acesso em 03.01.2022.

CAETANO, C., **Gestão de Defeitos**, Revista Engenharia de Software Magazine, Devmedia Editora, Rio de Janeiro: 1ª Edição, Ano I, p. 60- 67, publicado em nov. 2007, disponível em: <http://www.devmedia. com.br/post-8028-Revista-Engenharia-de-Software.html> acesso em 20.01. 2022.

CAETANO, C., **Gestão De Testes**, Revista Engenharia de Software-Magazine, Devmedia Editora, Rio de Janeiro: 3ª Edição, Ano I, p. 58- 66, publicado em jan. 2008, disponível em: <http://www.devmedia.com.br/post-8028-Revista-Engenharia-de-Software.html> acesso em 10 jan. 2022.

CAN. (2001). Disponível em: http://www.can-cia.org/. Acesso em 25.06.2021

CRISPIN, L. and GREGORY, J. 2009. *Agile Testing: A Practical Guide for Testers and Agile Teams*. Pearson Education

Degardin, V. Laly, P., Liénard, M. & Degauque, P., **Impulsive Noise on In-Vehicle Power Lines: Characterization and Impact on Communication Performance**, IEEE International Power line Communications and Its Applications Conference, ISPLC 2006, 26-29 March, (pp 222-226).

FlexRay Consortium. (2004). FlexRay Communication System, Protocol Specification, Version 2.0. Disponível em: www.flexray.com. Acesso em 25.06.2021.

Gomes, Humberto Vargas (2010), **Metodologia de Projeto de Software Embarcado Voltada ao Teste** / por Humberto Vargas. – Porto Alegre: Programa de Pós-Graduação em Computação, 2010, 104p.

Gouret, W., Nouvel, F., EL Zein, G**.** (2006). **Additional Network Using Automotive Powerline Communication**, IEEE International Power line Communications and Its Applications Conference, ISPLC 2006, 26-29 March, (pp 1087-1092).

Gouret, W., Nouvel, F., EL Zein, G. (2007), **High Data Rate Network Using Automotive Power Line**. IEEE International Power line Communications and Its Applications Conference, ISPLC 2007, 26-28 March. (pp 1-4).

HODEL, Kleber Nogueira (2018), **Planejamento e Estruturação de Software em Sistemas Embarcados Automotivos** / K.N. Hodel – versão corr. – São Paulo, 2018, 150p.

IEEE Standard 610-1990: ***IEEE Standard Glossary of Software Engineering Terminology***, IEEE Press.

IDB Forum. (2008). Disponível em: www.idbforum.org. Acesso em 15.06.2021.

IDB1394, (2008). 1394 **Automotive Presentation**, Max Bassler, Shenzen, China, 10 April 2008. Disponível em: http://www.idbforum.org. Acesso em 15.06.2021.

International Software Testing Qualifications Board (ISTQB), *Standard glossary of terms used in Software Testing* [online]. 2010

Johansson, K., Törngren, M., Nielsen, L. (2005), ***Handbook of Networked and Embedded Control Systems***, D. Hristu-Varsakelis and W. S. Levine, Eds. Boston, MA: Birkhäuser, 2005.

KOEN, B. V., 1985. *Definition of the Engineering Method*. American Society for Engineering Education.

Kopetz, H. at al. (2002). ***Specification of the TTP/A Protocol***. (Technical Report). Vienna, Austria, University. Technology.

LIN Consortium. (2003). LIN Specification Package, Revision 2.0. Disponível em: www.lin-subbus.org. Acesso em 16.06.2021.

MOST (2006). **MOST Core Compliance Test Specification V2.00, Technical Report**. Disponível em: www.mostcooperation.com/publications. Acesso em 22.06.2021.

MOST Cooperation, (2008). **MOST Specification Revision**. Disponível em: www.mostnet.de. Acesso em 22.06.2021.

Navet (2008), ***The Automotive Embedded Systems Handbook*, Industrial Information Technology series**, CRC Press / Taylor and Francis, ISBN 978-0849380266.

Navet, N., Song, Y-Q., Simonot-Lion, F. & Wilvert, C. (2005). **Trends in Automotive Communication Systems, *Proceedings of the IEEE***, *special issue on Industrial Communications Systems*. Invited paper, vol 93, n6, Junho (pp1204-1223).

Nolte, T. (2003). ***Reducing Pessimism and Increasing Flexibility in the Controller Area Network***. Mälardalen University Licentiate Thesis, May 2003 Department of Computer Science and Engineering Mälardalen University Västeras, Sweden, Copyright Thomas Nolte, Printed by Arkitektkopia, Västeras, Sweden Distribution .

Nolte,T. (2006). ***Share-Driven Scheduling of Embedded Networks***. Mälardalen University, Dissertation, May 2006. Department of Computer Science and Engineering Mälardalen University Västeras, Sweden, Printed by Arkitektkopia, Västeras, Sweden Distribution.

Nouvel, F., El Zein, G. & Citerne, J. (1994). ***Code division multiple access for an automotive area network over power-lines***. *Proceedings of IEEE Vehicular Technology Conf.*, junho (pp. 525–529).

PAULA, W. P. F., **Engenharia de Software – Fundamentos, Métodos e Padrões**. Rio de Janeiro: 2ª edição. GEN LTC, 2005.

PAULA, W. P. F., **Engenharia de Software – Fundamentos, Métodos e Padrões**. Rio de Janeiro: 3ª edição. GEN LTC, 2010.

PRESSMAN, R. S., **Engenharia de Software**. São Paulo: 3ª Edição. Makron Books, 1995.

ROMANO, S. M. V., **Gestão de Defeitos na Realização do Teste de Software**. Faculdade de Tecnologia de Praia Grande: Departamento de Análise e Desenvolvimento de Sistemas. Praia Grande, São Paulo.

SOMMERVILLE, I. **Engenharia de Software**, 8ª Edição. Pearson Education, 2007.

SOMMERVILLE, I. **Engenharia de Software**, 9ª Edição. Pearson Education, 2011.

VALENTE, M.T. Engenharia de Software Moderna: Princípios e Práticas para Desenvolvimento de Software com Produtividade, 1ª Edição, 2020.

Wazlawick, Raul Sidnei. Engenharia de software: conceitos e práticas. Rio de Janeiro: Elsevier, 2013.

APÊNDICE A – Casos de Teste Criados

TC01 - Conexão via Apple Carplay

[precondição_01] É necessário um iphone compatível com CarPlay com pelo menos iOS 15.0.

[precondição_02] Verifique se o iPhone está com Siri, *Wifi* e Bluetooth ativados

[passo_01] Verifique a presença "Apple CarPlay" na tela do veículo ou rack

[esperado] A opção Apple CarPlay está visível e ativa, assim como o bluetooth.

[passo _02] Clique em “conectar um novo dispositivo” e selecione Apple CarPlay

[esperado] As informações sobre o emparelhamento são exibidas

[passo _03] No iPhone, vá para Configurações -> Geral -> CarPlay

[esperado] A caixa de diálogo de emparelhamento do iPhone CarPlay exibe os veículos disponíveis para emparelhar

[passo _04] No iPhone, selecione o veículo para parear

[esperado] A tela exibe a solicitação de pareamento com código de autenticação e o iphone exibe a solicitação de emparelhamento com código de autenticação

[passo _05] confirme a solicitação de emparelhamento na tela e no iPhone

[esperado] A tela verificou a senha e voltou para a caixa de diálogo "Dispositivos móveis" e o iPhone emparelhado está listado como dispositivo Carplay

TC02 - Envio de Serviço Remoto via Smartphone

[precondição] O celular está pareado ao veículo/carro

[passo_01] No celular: Abra o aplicativo de serviços remotos e insira os dados de login.

[esperado] abre-se uma tela em que uma senha de 4 dígitos precisa ser criada

[passo_02] App: digite a senha duas vezes

[esperado] A página inicial do app é mostrada com veículo registrado e serviços remotos

[passo_03] App: Envie um serviço remoto (endereço de algum ponto de interesse)

[esperado] O endereço enviado é recebido no veículo/rack

TC03 - Verificação das informações de Trânsito em Tempo Real

[precondição_01] Informações de trânsito estão ativadas na tela do veículo

[precondição_02] Informações de trânsito estão disponíveis

[precondição_03] O veículo/rack possui boa recepção de sinal de dados de telefonia
[passo_01] Verifique se as informações de trânsito são exibidas na tela.
[Esperado] A situação de tráfego exibida deve corresponder à realidade:
*verde para fluxo livre
*amarelo para engarrafamento (cerca de 50% da velocidade máxima permitida)
*laranja para parar e ir (cerca de 25% da velocidade máxima permitida)
*vermelho para parada
*preto para fechamento de estrada

TC04 – Chamada de Emergência
[precondição _01] O veículo/rack está como modo direção ativo (simulação de motor ligado)
[precondição_02] O veículo/rack possui boa recepção de sinal de dados de telefonia
[passo_01] Pressione o botão de chamada de emergência no teto do veículo
[esperado] A informação da tela é alterada para a tela de chamada de emergência. Uma contagem regressiva de 5 segundos é iniciada. Após a contagem regressiva ser zero, a chamada é iniciada e bem-sucedida.
[passo_02] Peça ao agente da central de atendimento (que atendeu a chamada) para fazer um retorno de chamada após o término da mesma.
[esperado] O agente liga de volta em 3 minutos. A tela de chamada de emergência fica ativa novamente. Uma conexão telefônica com a central de atendimento é estabelecida automaticamente após um toque.
[passo_03] Fale com o agente
[esperado] verifique se é possível uma conversa com o agente
[passo_04] peça ao agente para encerrar a chamada.
[esperado] fim da chamada

TC05 - Funções Inteligentes-Regras de Climatização
[precondição_01] O veículo/rack deve estar “acordado”, porém com o motor está desligado.
[passo_01] Acesse o menu de aplicativos
[esperado] o Menu de aplicativos está aberto
[passo_02] Clique no recurso "Regulação do controle climático"
[esperado] O conteúdo da função deve ser aberto em menos de 10 segundos

[passo_03] Use o menu para definir o limite para a temperatura do ar-condicionado. Escolha uma temperatura significativamente inferior ou superior ao valor da temperatura externa.

[esperado] A temperatura é definida depois de clicar nela.

[passo_04] Selecione a intensidade de ventilação. Existem 7 níveis para a intensidade. Selecione o nível 3. Confirme a intensidade com um clique.

[passo_05] Desligue o veículo/rack e afaste-se dele por pelo menos 15 minutos.

[esperado] verifique se as regras selecionadas foram mantidas.

[passo_06] Repita todos os primeiros 5 passos para o controle do banco traseiro.

[esperado] conforme mencionado anteriormente.

TC06 – Chamadas Telefônicas em Paralelo

[precondição_01] Um dispositivo Android está conectado com Android Auto (espelhamento de tela).

[precondição_02] Um segundo telefone está conectado via Bluetooth (telefonia).

[passo_01] Inicie uma chamada telefônica ou receba uma chamada telefônica no segundo dispositivo móvel.

Não use o dispositivo Apple CarPlay para a chamada telefônica.

[esperado] A chamada telefônica Bluetooth ativa é exibida na tela.

[passo_02] Receba uma chamada telefônica no dispositivo Android Auto. Não use o segundo dispositivo Bluetooth para a chamada telefônica.

[esperado] um pop-up é exibido com a chamada telefônica recebida.

[passo_03] Aceite a chamada telefônica pressionando o botão “aceitar” no pop-up.

[esperado] A chamada telefônica Bluetooth termina, a tela principal muda da rack para a interface do usuário Android Auto e a chamada telefônica ativa é exibida.

TC07 - Navegação de Aprendizagem

[precondição_01] O sistema de navegação está totalmente disponível e o mapa é exibido na tela

[passo_01] Inicie uma rota para um destino específico

[esperado] A navegação está ativa

[passo_02] Dirija até este destino pela rota selecionada (preferida)

[esperado] Navegação é finalizada

[passo_03] Desligue o veículo/rack e afaste-se dele por pelo menos 15 minutos. Após isso e reinicie o sistema de navegação

[esperado] Navegação está ativa novamente

[passo_04] Repita os passos 1-4 pelo menos 6 vezes

[esperado] o sistema entenderá a rota preferida pelo usuário e dará uma sugestão de navegação na tentativa seguinte

[passo_05] Verifique se as informações estão disponíveis também nas telas do banco traseiro

[esperado] conforme passo_04.

TC08 - Trava de Velocidade – App de Notícias

[precondição_01] O menu Apps está aberto e o app News está visível.

[precondição_02] O veículo/rack está no modo de condução/movimento.

[passo_01] Abra o aplicativo de notícias e selecione um artigo.

[esperado] Por causa da trava de velocidade, apenas uma página do texto do artigo é mostrada. O texto não pode ser rolado. Um texto de sobreposição será exibido: “Para sua segurança: texto completo exibido apenas quando parado.**"**

[passo_02] Pare de dirigir (ou simular a direção) enquanto o artigo estiver aberto.

[esperado] O texto completo é mostrado e pode ser lido. O texto pode ser rolado.

[passo_03] Comece a dirigir (ou simular a direção) novamente.

[esperado] O texto é cortado novamente em apenas uma página. O texto de sobreposição é mostrado novamente.

TC09 – Comandos de Voz - Ventilação

[precondição_01] O veículo/rack está “acordado”

[precondição_02] O veículo está equipado com aquecimento do assento/ventilação do assento

[passo_01] Ative o diálogo de fala pressionando o botão de voz

[esperado] O diálogo de fala é iniciado e o sistema pergunta ao usuário qual é sua intenção.

[passo_02] Dê o comando "Definir aquecimento do banco para [Motorista / Passageiro] no nível [x]"

[esperado] A configuração desejada para o aquecimento do banco é ativada para o banco que foi solicitado.

[passo_03] Desative o aquecimento do banco

[esperado] O aquecimento está desativado

[passo_04] Repita todos os primeiros 3 passos para o controle do banco traseiro.
[esperado] conforme mencionado anteriormente.

TC10 – Sintonia de Estações de Rádio
[precondição_01] O veículo/rack deve estar “acordado”, porém com o motor está desligado.
[passo_01] Acesse o menu de mídia
[esperado] o Menu de mídia está aberto
[passo_02] Selecione rádio FM.
[esperado] Rádio FM está aberta
[passo_03] Selecione o campo de sintonia automática e aguarde até o processo finalizar
[esperado] as estações de rádio disponíveis na região estão salvas no sistema. Caso não haja estações disponíveis, nenhuma frequência deve ser armazenada no sistema.
[passo_06] Repita todos os primeiros 3 passos, mas nesse caso para rádio AM.
[esperado] as estações de rádio disponíveis na região estão salvas no sistema. Caso não haja estações disponíveis, nenhuma frequência deve ser armazenada no sistema.