Jobs

Questions

~~Automotive Ethernet~~
~~Communication Protocols - CAN,FlexRay~~
~~Read about ISO26262~~
~~TPT - Time Partition Testing~~
~~SiL,MiL,HiL~~
~~C Questions~~
~~Different Datatypes in C, VBA~~
~~Read about Embedded C~~
~~Difference between C and Embedded C~~
~~ISTQB Questions~~
~~Python~~
~~How to create Libraries in Python~~
~~Read about Functional Programming Concepts~~
~~Read about Scala,Clojure~~
~~Read about .NET Framework~~
~~Travis/Jenkins - Their Current Usage~~
~~Read about ADC, DAC and Succesive approximation Techniques~~
~~Difference between verification and validation~~
~~What is the MATLAB Version I am using?~~
~~Introducing Delays in Simulink~~

Information

Automotive

ADAS

https://en.wikipedia.org/wiki/Advanced_driver-assistance_systems

The US Department of Transportation’s National Highway Traffic Safety Administration (NHTSA) uses the Automation Levels[10] as defined by the Society of Automotive Engineers (SAE) to define a vehicle's capability from Level 0 (No Automation) to Level 5 (Full Automation).

Automation Levels From 0 to 5

• Adaptive cruise control (ACC)
• Glare-free high beam and pixel light
• Adaptive light control: swiveling curve lights
• Anti-lock braking system
• Automatic parking
• Automotive navigation system with typically GPS and TMC for providing up-to-date traffic information.
• Automotive night vision
• Blind spot monitor
• Collision avoidance system (Pre-crash system)
• Crosswind stabilization
• Cruise control
• Driver drowsiness detection
• Toyota Driver Monitoring System
• eCall
• Electric vehicle warning sounds used in hybrids and plug-in electric vehicles
• Electronic stability control (ESC)
• Emergency brake assist (EBA)
• Emergency driver assistant
• Forward collision warning
• Intersection assistant
• Hill descent control
• Intelligent speed adaptation or intelligent speed advice (ISA)
• Lane departure warning system
• Lane change assistance
• Night vision
• Parking sensor
• Pedestrian protection system
• Rain sensor
• Start-stop system
• Omniview technology
• Tire-pressure monitoring system
• Traffic-sign recognition
• Turning assistant
• Vehicular communication systems
• Wrong-way driving warning

Adaptive Cruise Control

https://en.wikipedia.org/wiki/Autonomous_cruise_control_system

Constantly monitor the Distance between your Vehicle and the vehicle before.
If it reaches a threshold, then automatically reduce the vehicle speed by altering the respective vehicle parameters

Lasers are not Effective **so RADARs are used instead**

Autonomous cruise control system

From Wikipedia, the free encyclopedia
(Redirected from Adaptive cruise control)

Autonomous cruise control (**ACC**; also called **adaptive cruise control**, **radar cruise control**, **traffic-aware cruise control** or **dynamic radar cruise control**) is an optional cruise control system for road vehicles that automatically adjusts the vehicle speed to maintain a safe distance from vehicles ahead.

Control is based on sensor information from on-board sensors. (No existing system makes use of satellite or roadside infrastructures nor of cooperative support from other vehicles.) Cooperative Adaptive Cruise Control (CACC) further extends the automation of navigation by using information gathered from fixed infrastructure such as satellites and roadside beacons, or mobile infrastructure such as reflectors or transmitters on the back of other vehicles.[citation needed]

Such systems may use a radar or laser sensor or a stereo camera setup allowing the vehicle to brake when it detects the car is approaching another vehicle ahead, then accelerate when traffic allows it to.

Schematic of Intelligent Cruise Control. The red car automatically follows the blue car.

Anti Lock Braking System

https://en.wikipedia.org/wiki/Anti-lock_braking_system

Monitor All Wheel Speeds
If 1 Wheel is faster than the other, then reduce the wheel speed by applying electronic brake using high pressure valve

The anti-lock brake controller is also known as the CAB (Controller Anti-lock Brake).[22]

Typically ABS includes a central electronic control unit (ECU), four wheel speed sensors, and at least two hydraulic valves within the brake hydraulics. The ECU constantly monitors the rotational speed of each wheel; if it detects the wheel rotating significantly slower than the speed of the vehicle, a condition indicative of impending wheel lock, it actuates the valves to reduce hydraulic pressure to the brake at the affected wheel, thus reducing the braking force on that wheel; the wheel then turns faster. Conversely, if the ECU detects a wheel turning significantly faster than the others, brake hydraulic pressure to the wheel is increased so the braking force is reapplied, slowing down the wheel. This process is repeated continuously and can be detected by the driver via brake pedal pulsation. Some anti-lock systems can apply or release braking pressure 15 times per second.[23] Because of this, the wheels of cars equipped with ABS are practically impossible to lock even during panic braking in extreme conditions.

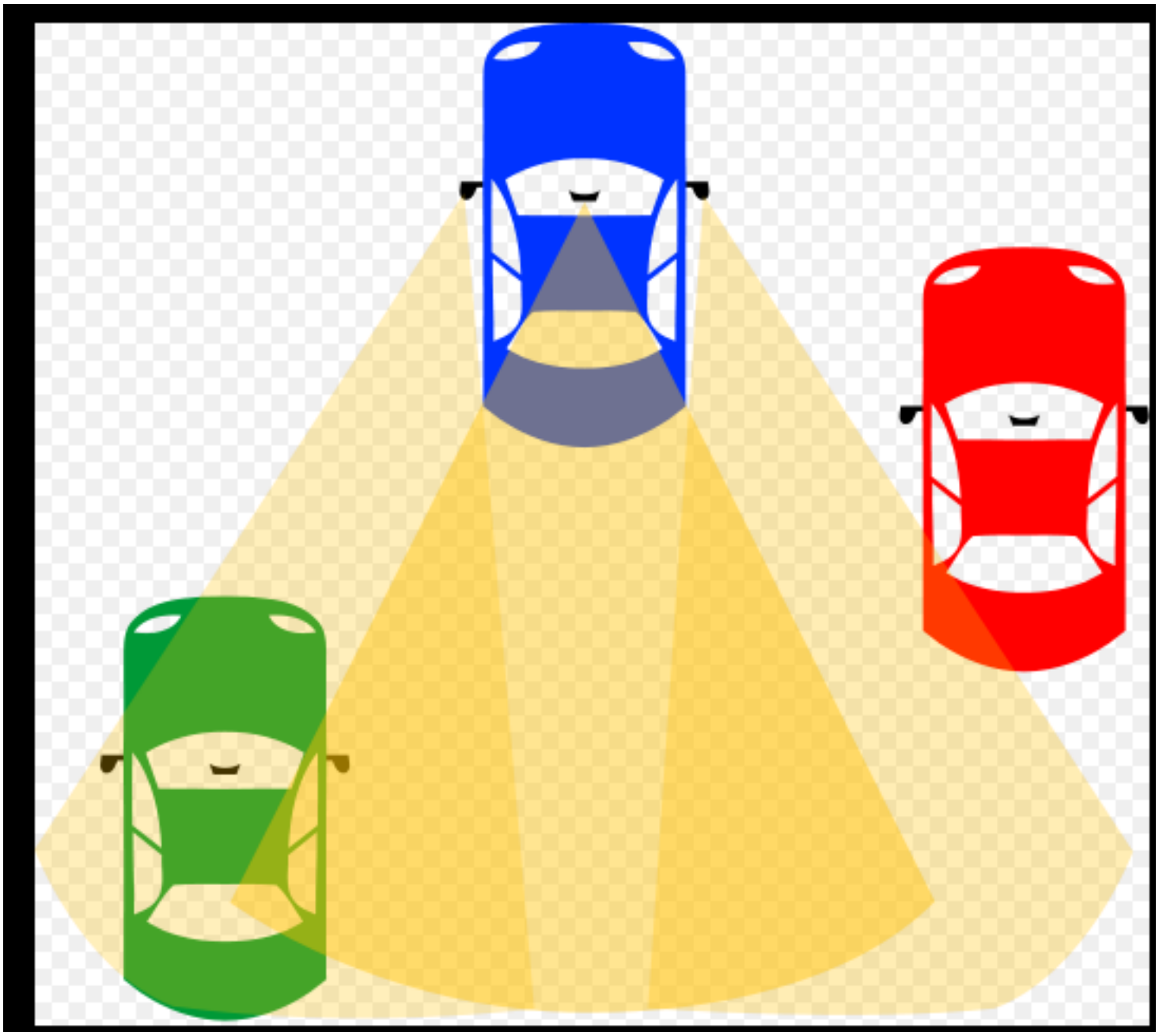
The ECU is programmed to disregard differences in wheel rotative speed below a critical threshold, because when the car is turning, the two wheels towards the center of the curve turn slower than the outer two. For this same reason, a differential is used in virtually all roadgoing vehicles.

If a fault develops in any part of the ABS, a warning light will usually be illuminated on the vehicle instrument panel, and the ABS will be disabled until the fault is rectified.

Modern ABS applies individual brake pressure to all four wheels through a control system of hub-mounted sensors and a dedicated micro-controller. ABS is offered or comes standard on most road vehicles produced today and is the foundation for electronic stability control systems, which are rapidly increasing in popularity due to the vast reduction in price of vehicle electronics over the years.[24]

Modern electronic stability control systems are an evolution of the ABS concept. Here, a minimum of two additional sensors are added to help the system work: these are a steering wheel angle sensor, and a gyroscopic sensor. The theory of operation is simple: when the gyroscopic sensor detects that the direction taken by the car does not coincide with what the steering wheel sensor reports, the ESC software will brake the necessary individual wheel(s) (up to three with the most sophisticated systems), so that the vehicle goes the way the driver intends. The steering wheel sensor also helps in the operation of Cornering Brake Control (CBC), since this will tell the ABS that wheels on the inside of the curve should brake more than wheels on the outside, and by how much.

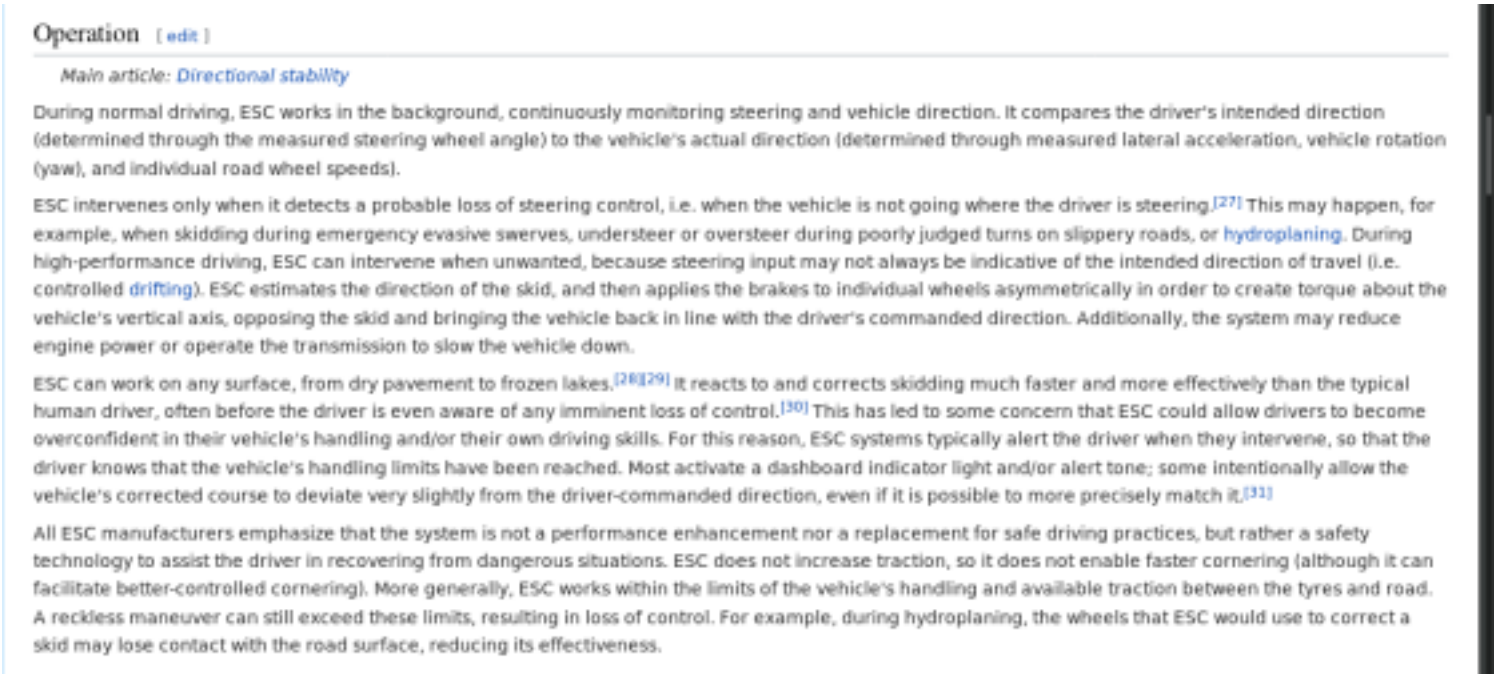
BlindSpot



Electronic Stability Control

https://en.wikipedia.org/wiki/Electronic_stability_control

Monitor Wheel Speed vs Steering Angle
Apply changes to the Wheels if there are any anomaly



Operation [edit]

Main article: Directional stability

During normal driving, ESC works in the background, continuously monitoring steering and vehicle direction. It compares the driver's intended direction (determined through the measured steering wheel angle) to the vehicle's actual direction (determined through measured lateral acceleration, vehicle rotation (yaw), and individual road wheel speeds).

ESC intervenes only when it detects a probable loss of steering control, i.e. when the vehicle is not going where the driver is steering.[27] This may happen, for example, when skidding during emergency evasive swerves, understeer or oversteer during poorly judged turns on slippery roads, or hydroplaning. During high-performance driving, ESC can intervene when unwanted, because steering input may not always be indicative of the intended direction of travel (i.e. controlled drifting). ESC estimates the direction of the skid, and then applies the brakes to individual wheels asymmetrically in order to create torque about the vehicle's vertical axis, opposing the skid and bringing the vehicle back in line with the driver's commanded direction. Additionally, the system may reduce engine power or operate the transmission to slow the vehicle down.

ESC can work on any surface, from dry pavement to frozen lakes.[28][29] It reacts to and corrects skidding much faster and more effectively than the typical human driver, often before the driver is even aware of any imminent loss of control.[30] This has led to some concern that ESC could allow drivers to become overconfident in their vehicle's handling and/or their own driving skills. For this reason, ESC systems typically alert the driver when they intervene, so that the driver knows that the vehicle's handling limits have been reached. Most activate a dashboard indicator light and/or alert tone; some intentionally allow the vehicle's corrected course to deviate very slightly from the driver-commanded direction, even if it is possible to more precisely match it.[31]

All ESC manufacturers emphasize that the system is not a performance enhancement nor a replacement for safe driving practices, but rather a safety technology to assist the driver in recovering from dangerous situations. ESC does not increase traction, so it does not enable faster cornering (although it can facilitate better-controlled cornering). More generally, ESC works within the limits of the vehicle's handling and available traction between the tyres and road. A reckless maneuver can still exceed these limits, resulting in loss of control. For example, during hydroplaning, the wheels that ESC would use to correct a skid may lose contact with the road surface, reducing its effectiveness.

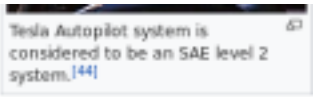
Levels - Autonomous Vehicles

Level 0 to 3 - Driver Required
Level 4,5 - Robotic

SAE (J3016) Autonomy Levels[49]

SAE Level	Name	Narrative definition		Execution of steering and acceleration/ deceleration	Monitoring of driving environment	Fallback performance of dynamic driving task	System capability (driving modes)
Human driver monitors the driving environment							
0	No Automation	The full-time performance by the human driver of all aspects of the dynamic driving task, even when "enhanced by warning or intervention systems"		Human driver	Human driver	Human driver	n/a
1	Drive Assistance	The driving mode-specific execution by a driver assistance system of "either steering or acceleration/deceleration"	using information about the driving environment and with the expectation that the human driver performs all remaining aspects of the dynamic driving task	Human driver and system			Some driving modes
2	Partial Automation	The driving mode-specific execution by one or more driver assistance systems of both steering and acceleration/deceleration		System			
Automated driving system monitors the driving environment							
3	Conditional Automation	The driving mode-specific performance by an automated driving system of all aspects of the dynamic driving task	with the expectation that the human driver will respond appropriately to a request to intervene	System	System	Human driver	Some driving modes
4	High Automation		even if a human driver does not respond appropriately to a request to intervene			System	Many driving modes
5	Full Automation		under all roadway and environmental conditions that can be managed by a human driver				All driving modes

Levels of driving automation [edit]



Tesla Autopilot system is considered to be an SAE level 2 system.[44]

In SAE's autonomy level definitions, "driving mode" means "a type of driving scenario with characteristic dynamic driving task requirements (e.g., expressway merging, high speed cruising, low speed traffic jam, closed-campus operations, etc.)"[49]

- Level 0: Automated system issues warnings and may momentarily intervene but has no sustained vehicle control.
- Level 1 ("hands on"): The driver and the automated system share control of the vehicle. Examples are Adaptive Cruise Control (ACC), where the driver controls steering and the automated system controls speed; and Parking Assistance, where steering is automated while speed is manual. The driver must be ready to retake full control at any time. Lane Keeping Assistance (LKA) Type II is a further example of level 1 self driving.
- Level 2 ("hands off"): The automated system takes full control of the vehicle (accelerating, braking, and steering). The driver must monitor the driving and be prepared to intervene immediately at any time if the automated system fails to respond properly. The shorthand "hands off" is not meant to be taken literally. In fact, contact between hand and wheel is often mandatory during SAE 2 driving, to confirm that the driver is ready to intervene.
- Level 3 ("eyes off"): The driver can safely turn their attention away from the driving tasks, e.g. the driver can text or watch a movie. The vehicle will handle situations that call for an immediate response, like emergency braking. The driver must still be prepared to intervene within some limited time, specified by the manufacturer, when called upon by the vehicle to do so. As an example, the 2018 Audi A8 Luxury Sedan was the first commercial car to claim to be capable of level 3 self driving. This particular car has a so-called Traffic Jam Pilot. When activated by the human driver, the car takes full control of all aspects of driving in slow-moving traffic at up to 60 kilometers per hour. The function works only on highways with a physical barrier separating one stream of traffic from oncoming traffic.
- Level 4 ("mind off"): As level 3, but no driver attention is ever required for safety, i.e. the driver may safely go to sleep or leave the driver's seat. Self driving is supported only in limited spatial areas (geofenced) or under special circumstances, like traffic jams. Outside of these areas or circumstances, the vehicle must be able to safely abort the trip, i.e. park the car, if the driver does not retake control.
- Level 5 ("steering wheel optional"): No human intervention is required. An example would be a robotic taxi.

Traction Control System

https://en.wikipedia.org/wiki/Traction_control_system

> Intervention consists of one or more of the following:
>
> - Brake force applied to one or more wheels
> - Reduction or suppression of spark sequence to one or more cylinders
> - Reduction of fuel supply to one or more cylinders
> - Closing the throttle, if the vehicle is fitted with drive by wire throttle
> - In turbocharged vehicles, a boost control solenoid is actuated to reduce boost and therefore engine power.

> ## Operation [edit]
>
> When the traction control computer (often incorporated into another control unit, such as the ABS module) detects one or more driven wheels spinning significantly faster than another, it invokes the ABS electronic control unit to apply brake friction to wheels spinning with lessened traction. Braking action on slipping wheel(s) will cause power transfer to wheel axle(s) with traction due to the mechanical action within the differential. All-wheel drive (AWD) vehicles often have an electronically controlled coupling system in the transfer case or transaxle engaged (active part-time AWD), or locked-up tighter (in a true full-time set up driving all wheels with some power all the time) to supply non-slipping wheels with torque.
>
> This often occurs in conjunction with the powertrain computer reducing available engine torque by electronically limiting throttle application and/or fuel delivery, retarding ignition spark, completely shutting down engine cylinders, and a number of other methods, depending on the vehicle and how much technology is used to control the engine and transmission. There are instances when traction control is undesirable, such as trying to get a vehicle unstuck in snow or mud. Allowing one wheel to spin can propel a vehicle forward enough to get it unstuck, whereas both wheels applying a limited amount of power will not produce the same effect. Many vehicles have a traction control shut-off switch for such circumstances.

Components of traction control [edit]

Generally, the main hardware for traction control and ABS are mostly the same. In many vehicles traction control is provided as an additional option to ABS.

- Each wheel is equipped with a sensor which senses changes in its speed due to loss of traction.
- The sensed speed from the individual wheels is passed on to an electronic control unit (ECU).
- The ECU processes the information from the wheels and initiates braking to the affected wheels via a cable connected to an automatic traction control (ATC) valve.

In all vehicles, traction control is automatically started when the sensors detect loss of traction at any of the wheels.

Miscellaneous

TPMS, BCM, Cluster

Tire Pressure Monitoring System

https://en.wikipedia.org/wiki/Tire-pressure_monitoring_system

Tire-pressure monitoring system

From Wikipedia, the free encyclopedia

A **tire-pressure monitoring system** (**TPMS**) is an electronic system designed to monitor the air pressure inside the pneumatic tires on various types of vehicles. TPMS report real-time tire-pressure information to the driver of the vehicle, either via a gauge, a pictogram display, or a simple low-pressure warning light. TPMS can be divided into two different types – direct (dTPMS) and indirect (iTPMS). TPMS are provided both at an OEM (factory) level as well as an aftermarket solution. The target of a TPMS is avoiding traffic accidents, poor fuel economy, and increased tire wear due to under-inflated tires through early recognition of a hazardous state of the tires

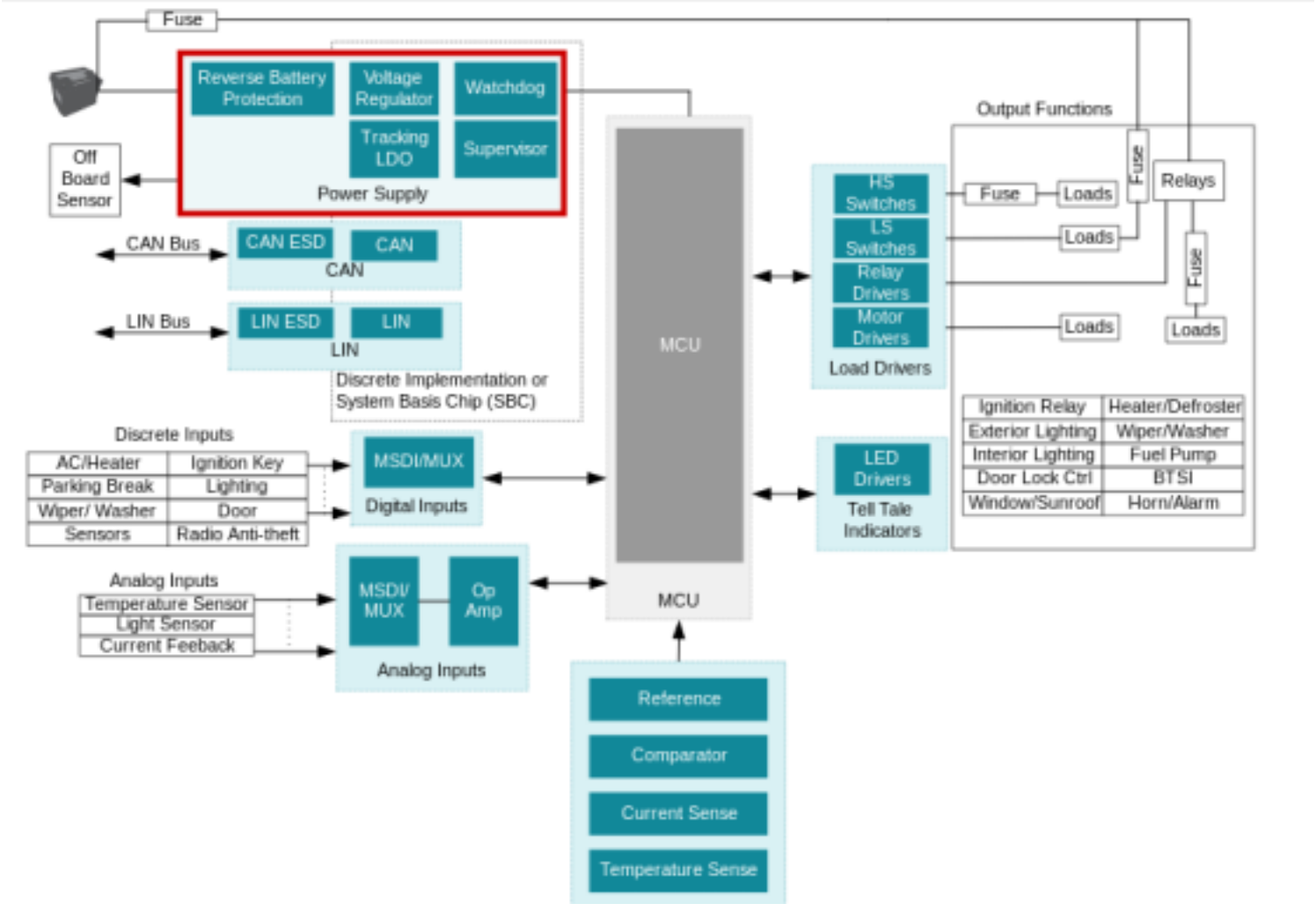
Body Control Module

https://en.wikipedia.org/wiki/Body_control_module

Body control module

From Wikipedia, the free encyclopedia

In automotive electronics, **body control module** or 'body computer' is a generic term for an electronic control unit responsible for monitoring and controlling various electronic accessories in a vehicle's body. Typically in a car the BCM controls the power windows, power mirrors, air conditioning, immobilizer system, central locking, etc. The BCM communicates with other on-board computers via the car's vehicle bus, and its main application is controlling load drivers – actuating relays that in turn perform actions in the vehicle such as locking the doors or dimming the salon overhead lamp.



Instrument Cluster

https://en.wikipedia.org/wiki/Electronic_instrument_cluster

Electronic instrument cluster

From Wikipedia, the free encyclopedia

> *"Digital dash" redirects here. For the business tool, see Dashboard (disambiguation).*

In an automobile, an **electronic instrument cluster**, **digital instrument panel** or *digital dash* for short, is a set of instrumentation, including the speedometer, that is displayed with a digital readout rather than with the traditional analog gauges. Many refer to it simply as a *digital speedometer.*

PowerTrain

https://en.wikipedia.org/wiki/Powertrain

In a wider sense, the powertrain includes all of its components used to transform stored (chemical, solar, nuclear, kinetic, potential, etc.) energy into kinetic energy for propulsion purposes.
This includes the utilization of multiple power sources and non–wheel-based vehicles.

In a motor vehicle, the term **powertrain** or **powerplant** describes the main components that generate power and deliver it to the road surface, water, or air. This includes the engine, transmission, drive shafts, differentials, and the final drive (drive wheels, continuous track as in military tanks or caterpillar tractors, propeller, etc.). More recently in hybrid powertrains the battery, the electric motor and the control algorithm are also seen as elements of the powertrain.

A motor vehicle's **driveline** or drivetrain consists of the parts of the powertrain excluding the engine. It is the portion of a vehicle, after the prime mover, that changes depending on whether a vehicle is front-wheel, rear-wheel, or four-wheel drive, or less-common six-wheel or eight-wheel drive.

In a wider sense, the powertrain includes all of its components used to transform stored (chemical, solar, nuclear, kinetic, potential, etc.) energy into kinetic energy for propulsion purposes. This includes the utilization of multiple power sources and non–wheel-based vehicles.

Engine

https://en.wikipedia.org/wiki/Internal_combustion_engine

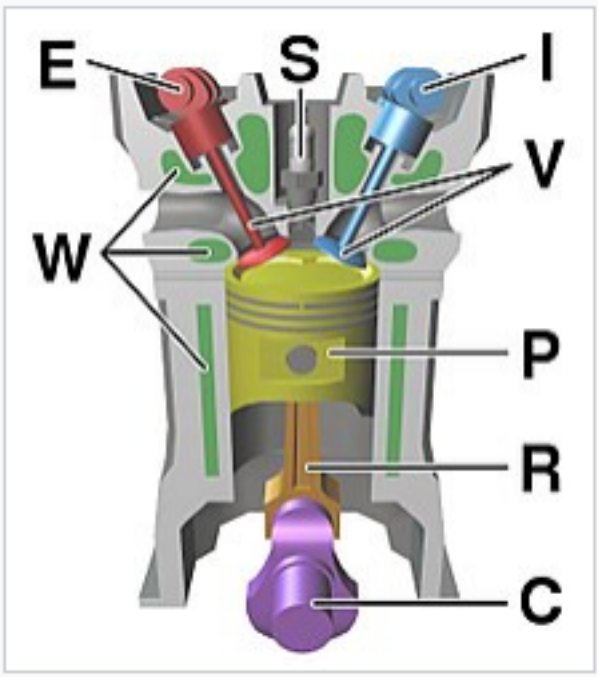


Diagram of a cylinder as found in 4-stroke gasoline engines.:
C – crankshaft.
E – exhaust camshaft.
I – inlet camshaft.
P – piston.
R – connecting rod.
S – spark plug.
V – valves. red: exhaust, blue: intake.
W – cooling water jacket.
gray structure – engine block.

4-stroke engines [edit]

Main article: 4-stroke engine

The *top dead center* (TDC) of a piston is the position where it is nearest to the valves; *bottom dead center* (BDC) is the opposite position where it is furthest from them. A *stroke* is the movement of a piston from TDC to BDC or vice versa, together with the associated process. While an engine is in operation, the crankshaft rotates continuously at a nearly constant speed. In a 4-stroke ICE, each piston experiences 2 strokes per crankshaft revolution in the following order. Starting the description at TDC, these are:[8][9]

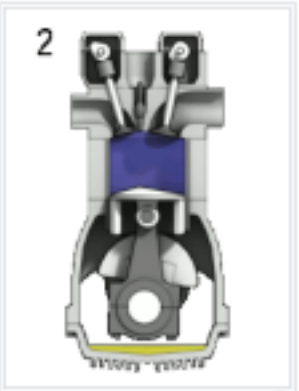
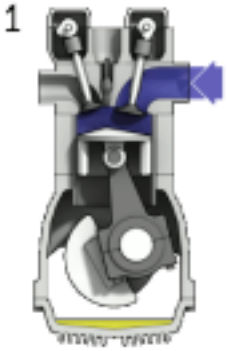


Diagram showing the operation of a 4-stroke SI engine. Labels:
1 - Induction
2 - Compression
3 - Power
4 - Exhaust

1. **Intake**, **induction** or **suction**: The intake valves are open as a result of the cam lobe pressing down on the valve stem. The piston moves downward increasing the volume of the combustion chamber and allowing air to enter in the case of a CI engine or an air fuel mix in the case of SI engines that do not use direct injection. The air or air-fuel mixture is called the *charge* in any case.
2. **Compression**: In this stroke, both valves are closed and the piston moves upward reducing the combustion chamber volume which reaches its minimum when the piston is at TDC. The piston performs work on the charge as it is being compressed; as a result its pressure, temperature and density increase; an approximation to this behavior is provided by the ideal gas law. Just before the piston reaches TDC, ignition begins. In the case of a SI engine, the spark plug receives a high voltage pulse that generates the spark which gives it its name and ignites the charge. In the case of a CI engine the fuel injector quickly injects fuel into the combustion chamber as a spray; the fuel ignites due to the high temperature.
3. **Power** or **working stroke**: The pressure of the combustion gases pushes the piston downward, generating more work than it required to compress the charge. Complementary to the compression stroke, the combustion gases expand and as a result their temperature, pressure and density decreases. When the piston is near to BDC the exhaust valve opens. The combustion gases expand irreversibly due to the leftover pressure—in excess of back pressure, the gauge pressure on the exhaust port—; this is called the *blowdown*.
4. **Exhaust**: The exhaust valve remains open while the piston moves upward expelling the combustion gases. For naturally aspirated engines a small part of the combustion gases may remain in the cylinder during normal operation because the piston does not close the combustion chamber completely; these gases dissolve in the next charge. At the end of this stroke, the exhaust valve closes, the intake valve opens, and the sequence repeats in the next cycle. The intake valve may open before the exhaust valve closes to allow better scavenging.

A **four-stroke** (also **four-cycle**) **engine** is an internal combustion (IC) engine in which the piston completes four separate strokes while turning the crankshaft. A stroke refers to the full travel of the piston along the cylinder, in either direction. The four separate strokes are termed:



Four-stroke cycle used in gasoline/petrol engines. **1** = Intake, **2** = Compression, **3** = Power, **4** = Exhaust. The right blue side is the intake port and the left brown side is the exhaust port. The cylinder wall is a thin sleeve surrounding the piston head which creates a space for the

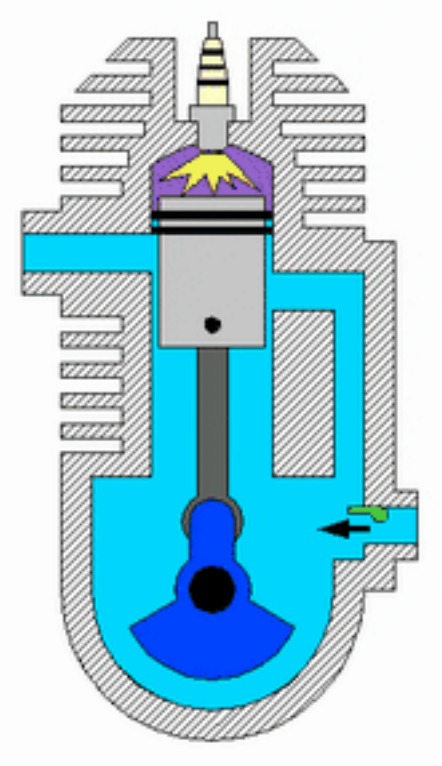
1. Intake: also known as induction or suction. This stroke of the piston begins at top dead center (T.D.C.) and ends at bottom dead center (B.D.C.). In this stroke the intake valve must be in the open position while the piston pulls an air-fuel mixture into the cylinder by producing vacuum pressure into the cylinder through its downward motion. The piston is moving down as air is being sucked in by the downward motion against the piston
2. Compression: This stroke begins at B.D.C, or just at the end of the suction stroke, and ends at T.D.C. In this stroke the piston compresses the air-fuel mixture in preparation for ignition during the power stroke (below). Both the intake and exhaust valves are closed during this stage.
3. Combustion: also known as power or ignition This is the start of the second revolution of the four stroke cycle. At this point the crankshaft has completed a full 360 degree revolution. While the piston is at T.D.C. (the end of the compression stroke) the compressed air-fuel mixture is ignited by a spark plug (in a gasoline engine) or by heat generated by high compression (diesel engines), forcefully returning the piston to B.D.C. This stroke produces mechanical work from the engine to turn the crankshaft.
4. Exhaust: also known as outlet. During the *exhaust* stroke, the piston once again returns from B.D.C. to T.D.C. while the exhaust valve is open. This action expels the spent air-fuel mixture through the exhaust valve.

Contents [hide]

1 History

g cycle

2 Stroke Engines

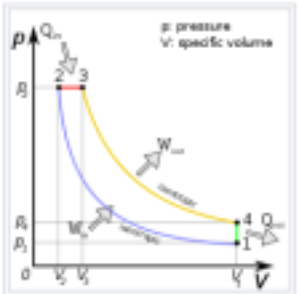


PV Diagram in Engine

Diesel cycle [edit]

Main article: Diesel cycle

Most truck and automotive diesel engines use a cycle reminiscent of a four-stroke cycle, but with compression heating causing ignition, rather than needing a separate ignition system. This variation is called the diesel cycle. In the diesel cycle, diesel fuel is injected directly into the cylinder so that combustion occurs at constant pressure, as the piston moves.



P-v Diagram for the ideal Diesel cycle. The cycle follows the numbers 1–4 in clockwise direction.

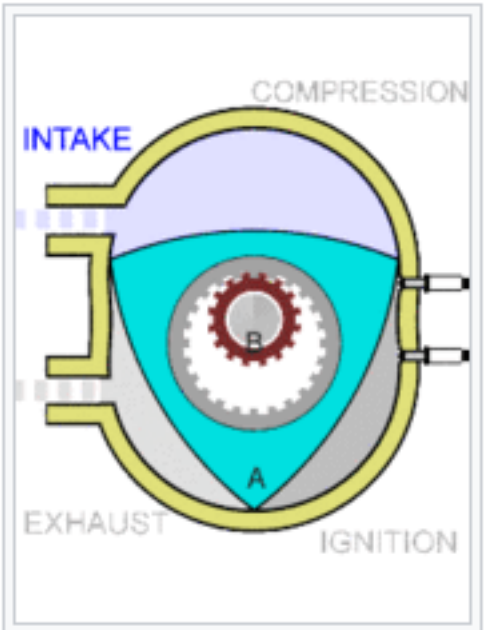
Otto cycle [edit]

Otto cycle is the typical cycle for most of the cars internal combustion engines, that work using gasoline as a fuel. Otto cycle is exactly the same one that was described for the four-stroke engine. It consists of the same major steps: intake, compression, ignition, expansion and exhaust.

Five-stroke engine [edit]

In 1879, Nikolaus Otto manufactured and sold a double expansion engine (the double and triple expansion principles had ample usage in steam engines), with two small cylinders at both sides of a low-pressure larger cylinder, where a second expansion of exhaust stroke gas took place; the owner returned it, alleging poor performance. In 1906, the concept was incorporated in a car built by EHV (Eisenhuth Horseless Vehicle Company) CT, USA;[22] and in the 21st century Ilmor designed and successfully tested a 5-stroke double expansion internal combustion engine, with high power output and low SFC (Specific Fuel Consumption).[23]

Wankel Engine

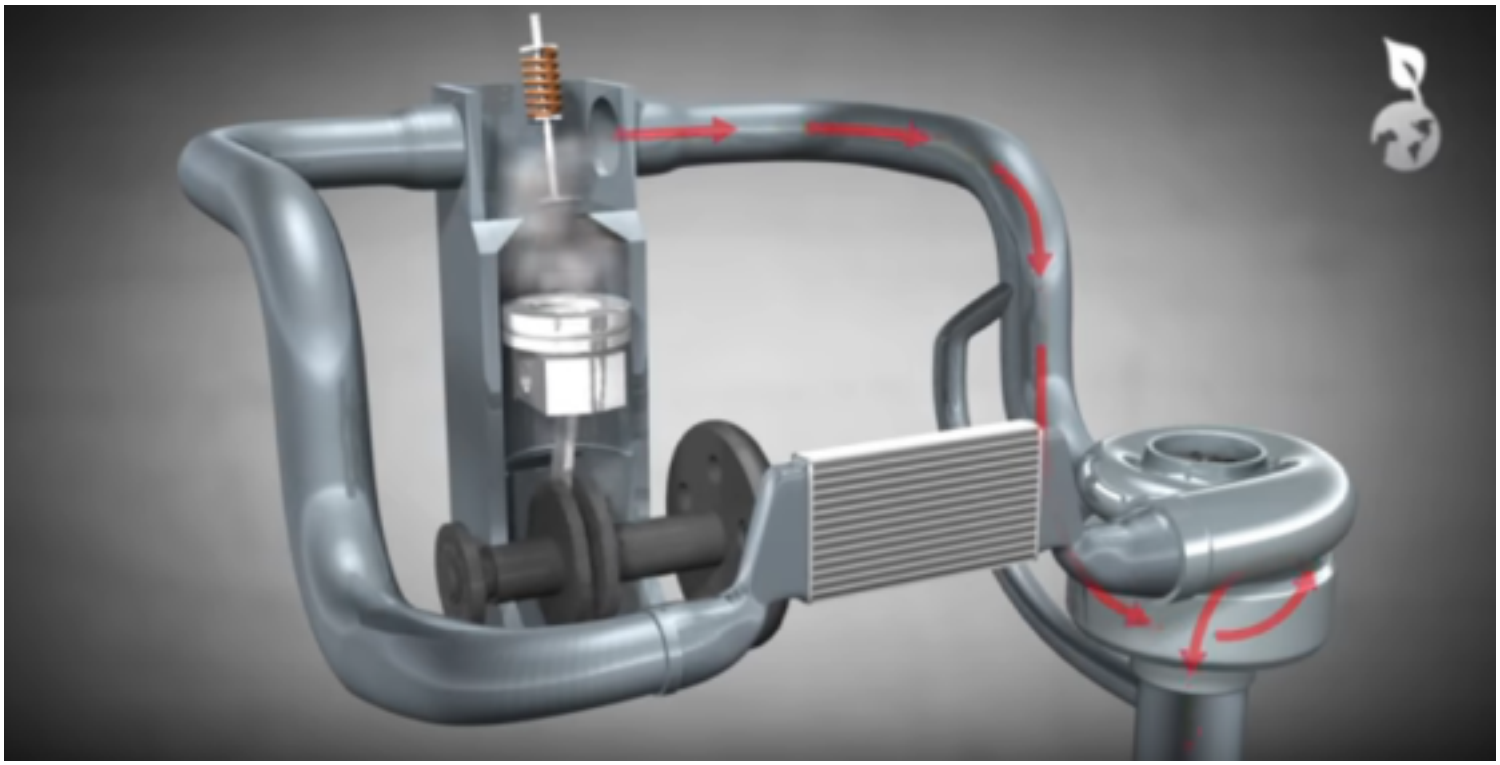


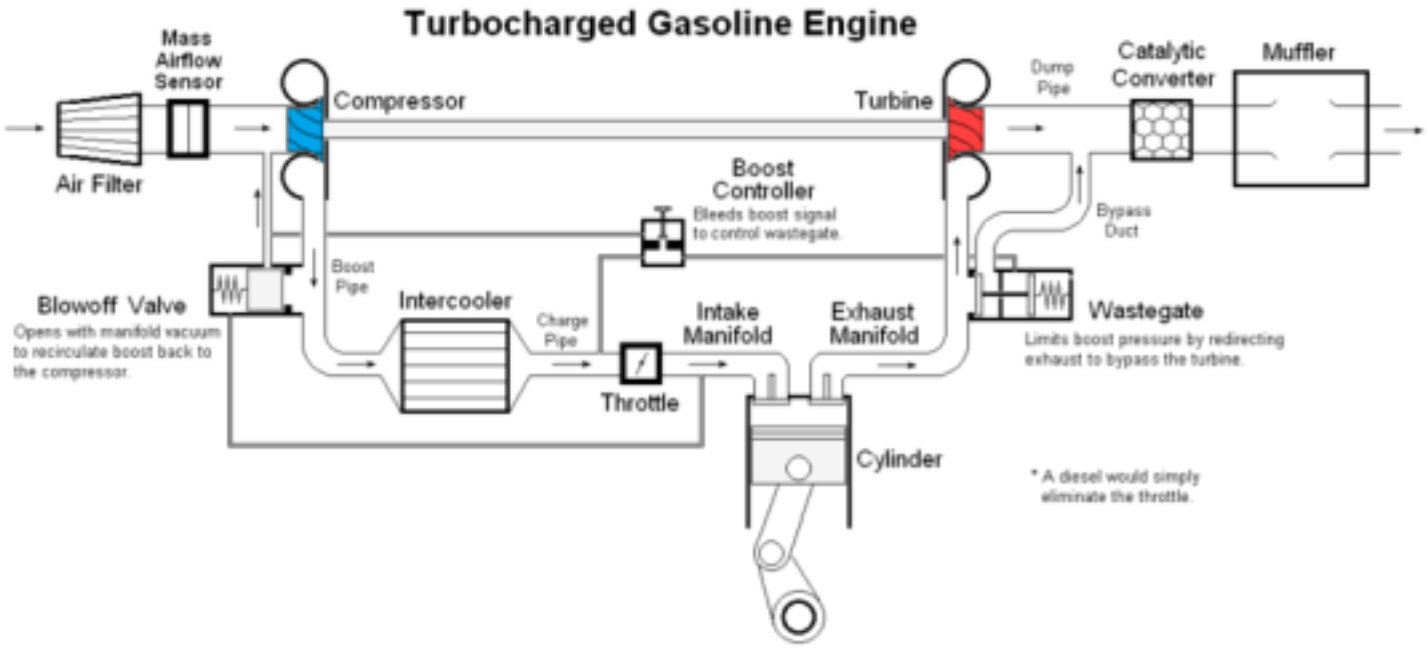
The Wankel rotary cycle. The shaft turns three times for each rotation of the rotor around the lobe and once for each orbital revolution around the eccentric shaft.

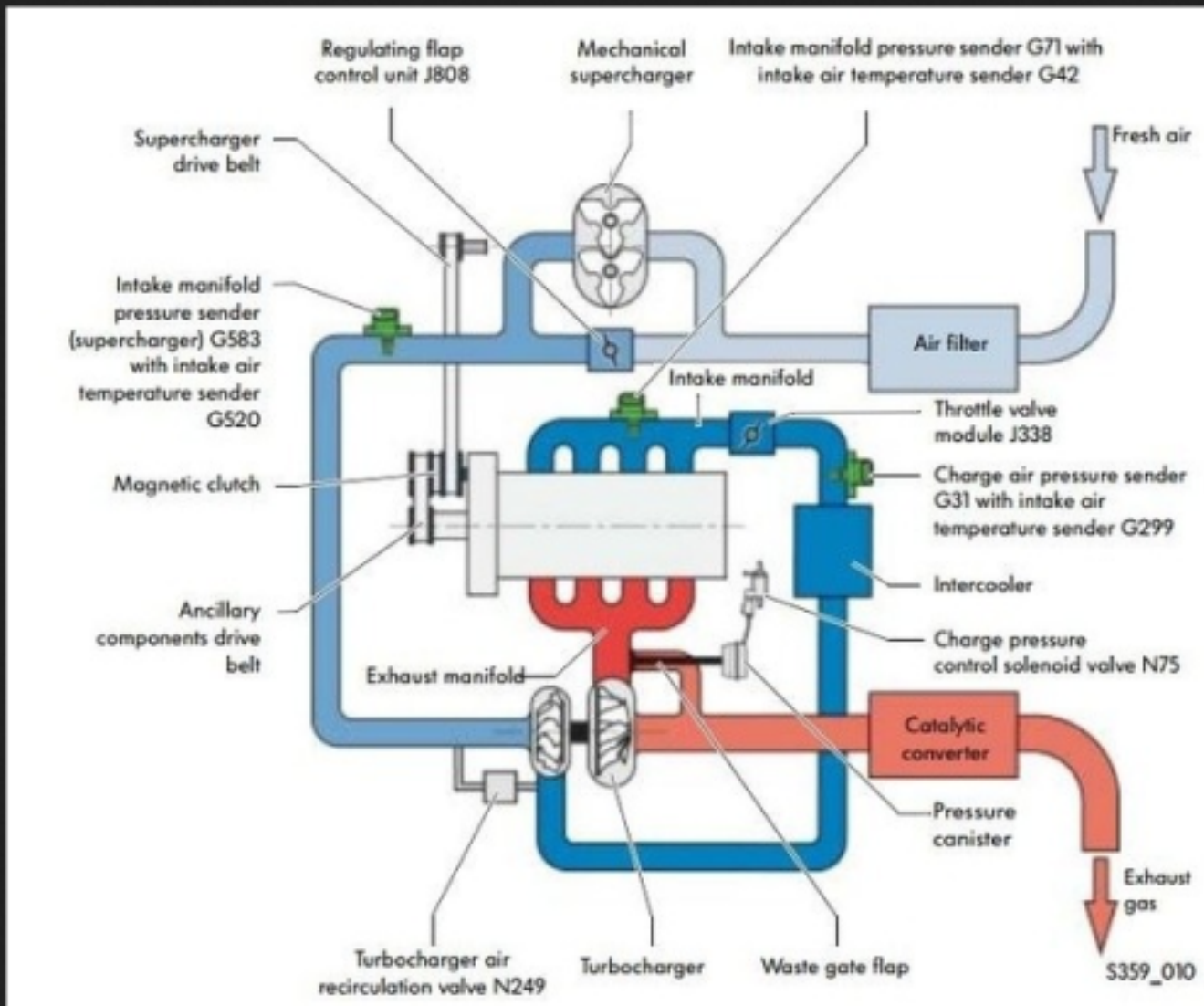
TurboCharger

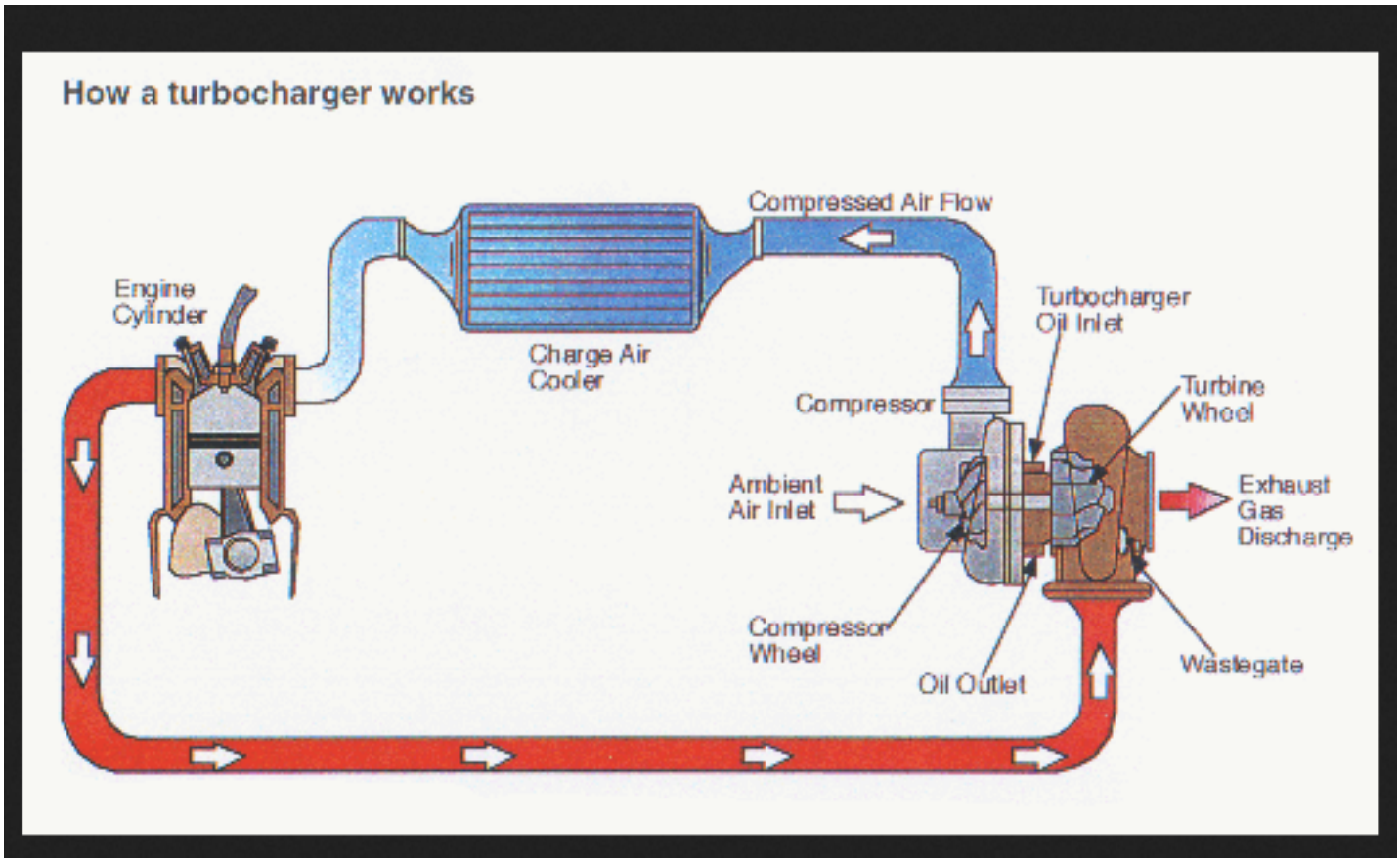
Used to Push more air in to the Intake by reusing the Exhaust Gases
Helps for Better Efficiency of the Engine

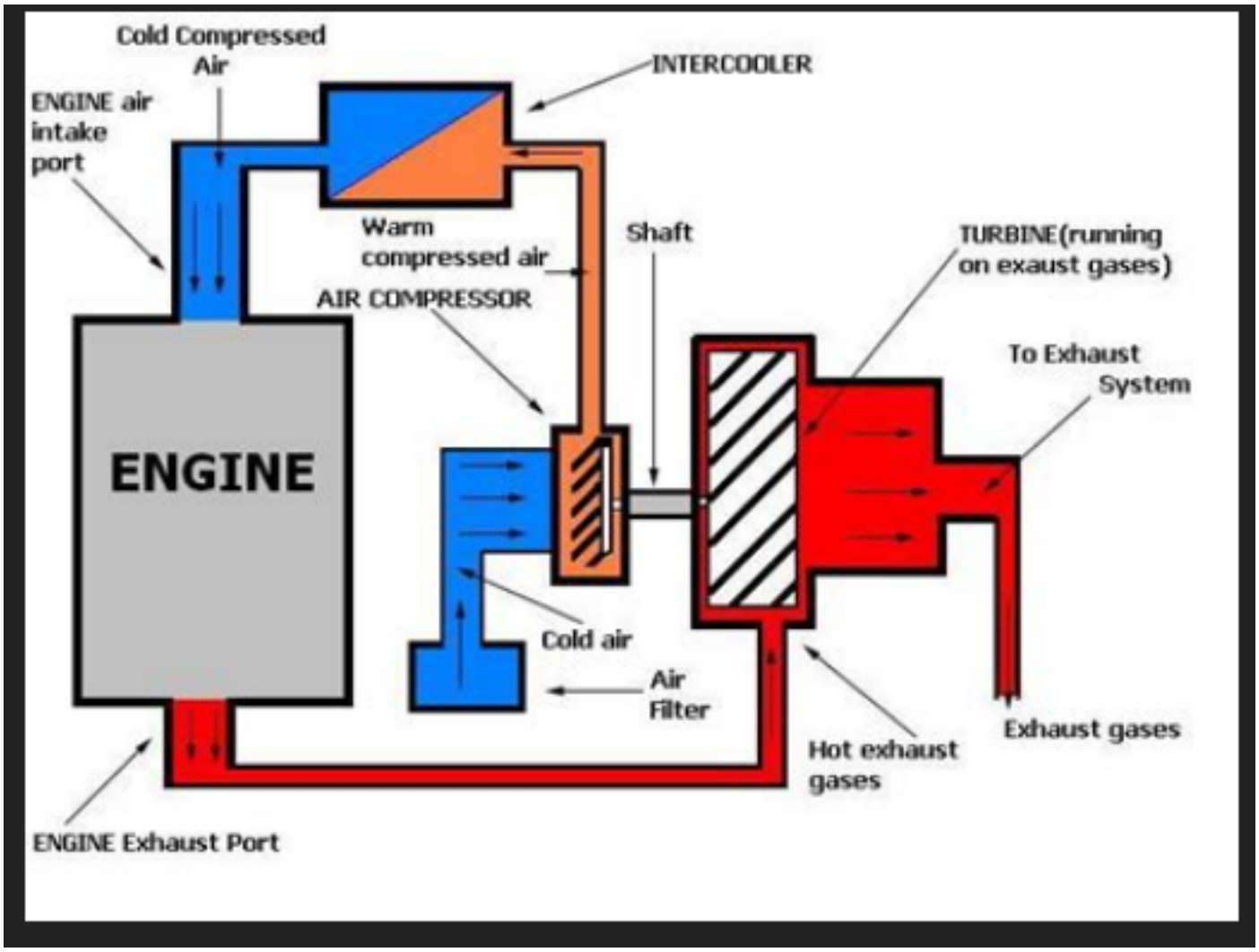
How a Turbocharger Works Animation
https://www.youtube.com/watch?v=K8aCL56jZX4




Turbocharged Gasoline Engine
Mass Airflow Sensor
Air Filter
Compressor
Turbine
Dump Pipe
Catalytic Converter
Muffler
Boost Controller
Bleeds boost signal to control wastegate.
Bypass Duct
Boost Pipe
Blowoff Valve
Opens with manifold vacuum to recirculate boost back to the compressor.
Intercooler
Charge Pipe
Intake Manifold
Exhaust Manifold
Wastegate
Limits boost pressure by redirecting exhaust to bypass the turbine.
Throttle
Cylinder
* A diesel would simply eliminate the throttle.


Regulating flap control unit J808
Mechanical supercharger
Intake manifold pressure sender G71 with intake air temperature sender G42
Fresh air
Supercharger drive belt
Intake manifold pressure sender (supercharger) G583 with intake air temperature sender G520
Air filter
Intake manifold
Throttle valve module J338
Magnetic clutch
Charge air pressure sender G31 with intake air temperature sender G299
Intercooler
Ancillary components drive belt
Charge pressure control solenoid valve N75
Exhaust manifold
Catalytic converter
Pressure canister
Exhaust gas
Turbocharger air recirculation valve N249
Turbocharger
Waste gate flap
S359_010

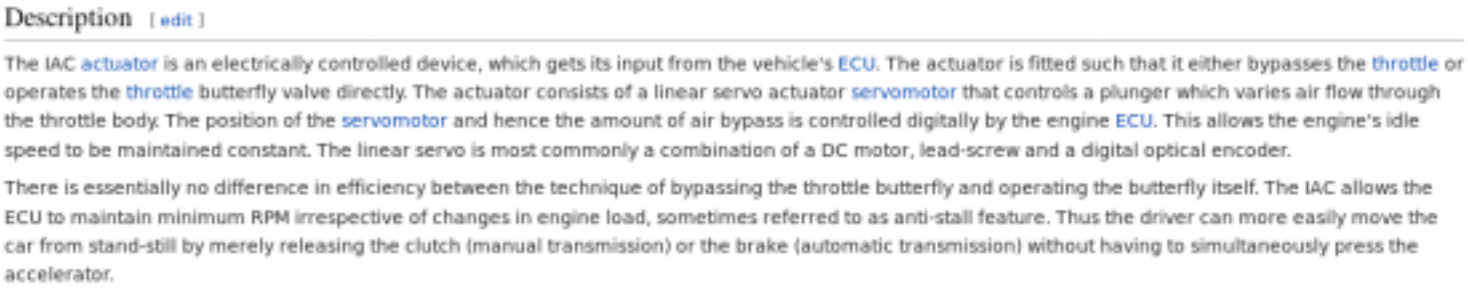




Knocking in Engines

Idle Speed Control

https://en.wikipedia.org/wiki/Idle_air_control_actuator



Description [edit]

The IAC actuator is an electrically controlled device, which gets its input from the vehicle's ECU. The actuator is fitted such that it either bypasses the throttle or operates the throttle butterfly valve directly. The actuator consists of a linear servo actuator servomotor that controls a plunger which varies air flow through the throttle body. The position of the servomotor and hence the amount of air bypass is controlled digitally by the engine ECU. This allows the engine's idle speed to be maintained constant. The linear servo is most commonly a combination of a DC motor, lead-screw and a digital optical encoder.

There is essentially no difference in efficiency between the technique of bypassing the throttle butterfly and operating the butterfly itself. The IAC allows the ECU to maintain minimum RPM irrespective of changes in engine load, sometimes referred to as anti-stall feature. Thus the driver can more easily move the car from stand-still by merely releasing the clutch (manual transmission) or the brake (automatic transmission) without having to simultaneously press the accelerator.

Based on Different Vehicle Conditions, the Idle Speed has to be maintained so that the user does not feel
the jerk. For Example, if the user switches on Additional Load say AC when the engine is in idle condition,
this exerts more load on the Engine which means more torque to the generated to compensate.
Such Mechanisms are handled in the Idle Speed Control

Torque Distribution/Torque Reservation

Based on Different ECU Inputs, how torque should be distributed

1. From ECUs like TCM, ESP & ART
2. Torque Minimum Mode, Torque Maximal Mode, Torque Intervention Mode
3. Based on the Modes how Torque is requested there will be either an additional Torque Generated and the respective request goes to the air path and torque is generated
4. If a minimal request is received, then again it affects the air path and fuel path for reduced torque

Also the situations in which how the torque should be distributed to the wheels based on the vehicle
dynamics

http://www.awdwiki.com/en/torque+split+ratio/

Hybrid Vehicle

https://en.wikipedia.org/wiki/Hybrid_vehicle

Types of Hybrid Vehicle

https://en.wikipedia.org/wiki/Hybrid_vehicle_drivetrain

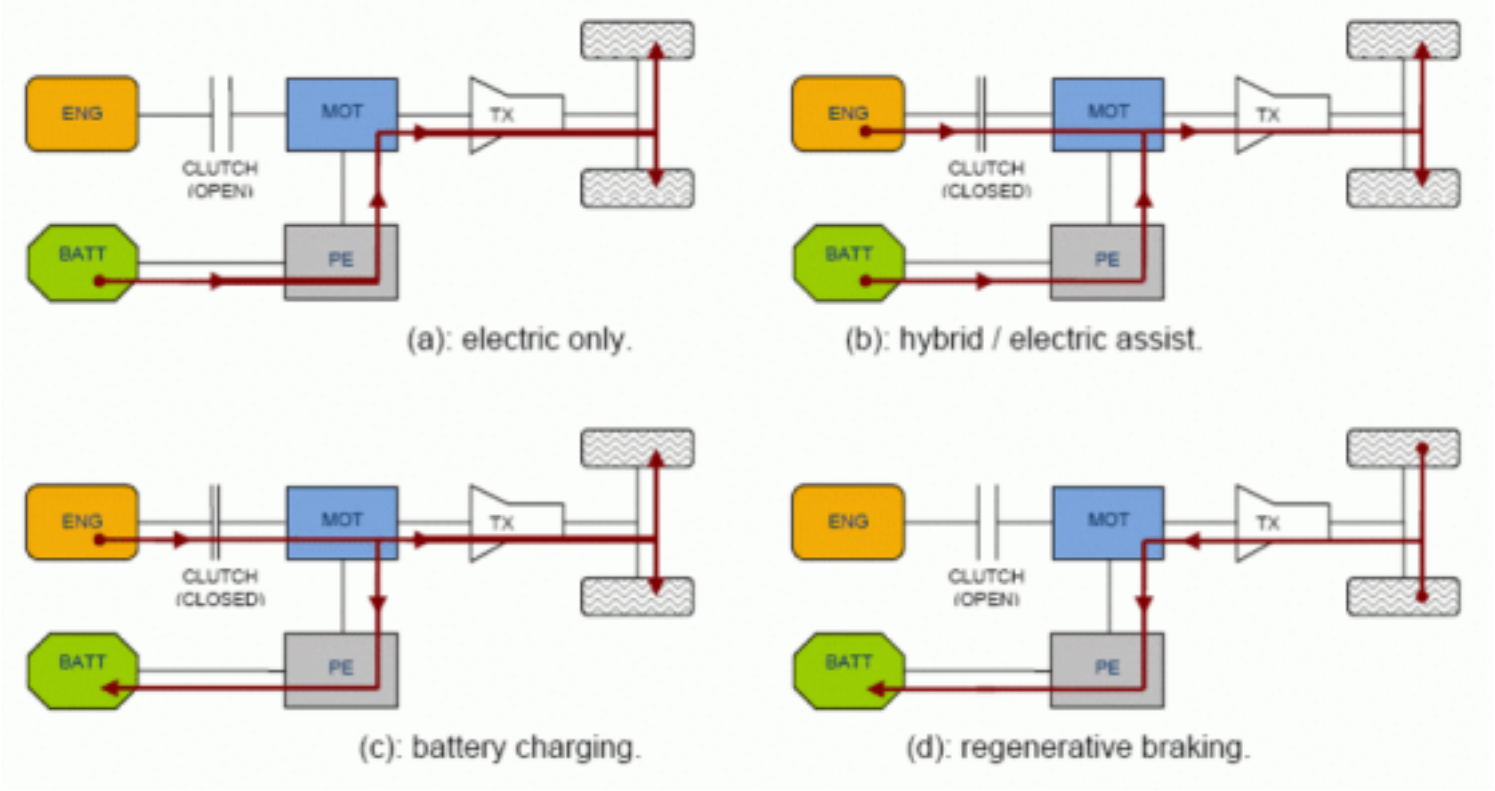
1 Types by design
- 1.1 Parallel hybrid
- 1.2 Series hybrid
- 1.3 Electric traction motors
- 1.4 Power-split or series-parallel hybrid

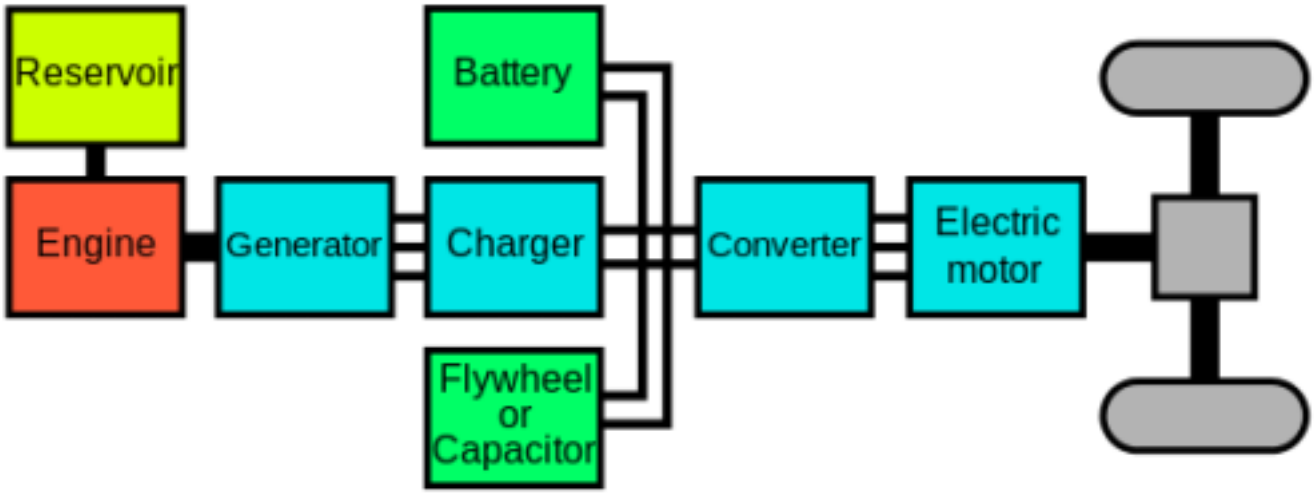
2 Types by degree of hybridization
- 2.1 Micro hybrids
- 2.2 Mild hybrids
- 2.3 Full hybrids
- 2.4 Plug-in hybrid

3 Types by power source
- 3.1 Electric-internal combustion engine hybrid
- 3.2 Electric-fuel cell hybrid
- 3.3 Internal combustion engine-hydraulic hybrid
- 3.4 Internal combustion engine-pneumatic
- 3.5 Human power-environmental power

Type	Start-stop system	Regenerative braking Electric boost	Charge-depleting mode	Rechargeable
Micro hybrid	Yes	No	No	No
Mild hybrid	Yes	Yes	No	No
Full hybrid	Yes	Yes	Yes	No
Plug-in hybrid	Yes	Yes	Yes	Yes

Hybrid Mode Operation


ENG
MOT
TX
CLUTCH
(OPEN)
BATT
PE
(a): electric only.
ENG
MOT
TX
CLUTCH
(CLOSED)
BATT
PE
(b): hybrid / electric assist.
ENG
MOT
TX
CLUTCH
(CLOSED)
BATT
PE
(c): battery charging.
ENG
MOT
TX
CLUTCH
(OPEN)
BATT
PE
(d): regenerative braking.


Reservoir
Battery
Engine
Generator
Charger
Converter
Electric motor
Flywheel or Capacitor

Torque Converter

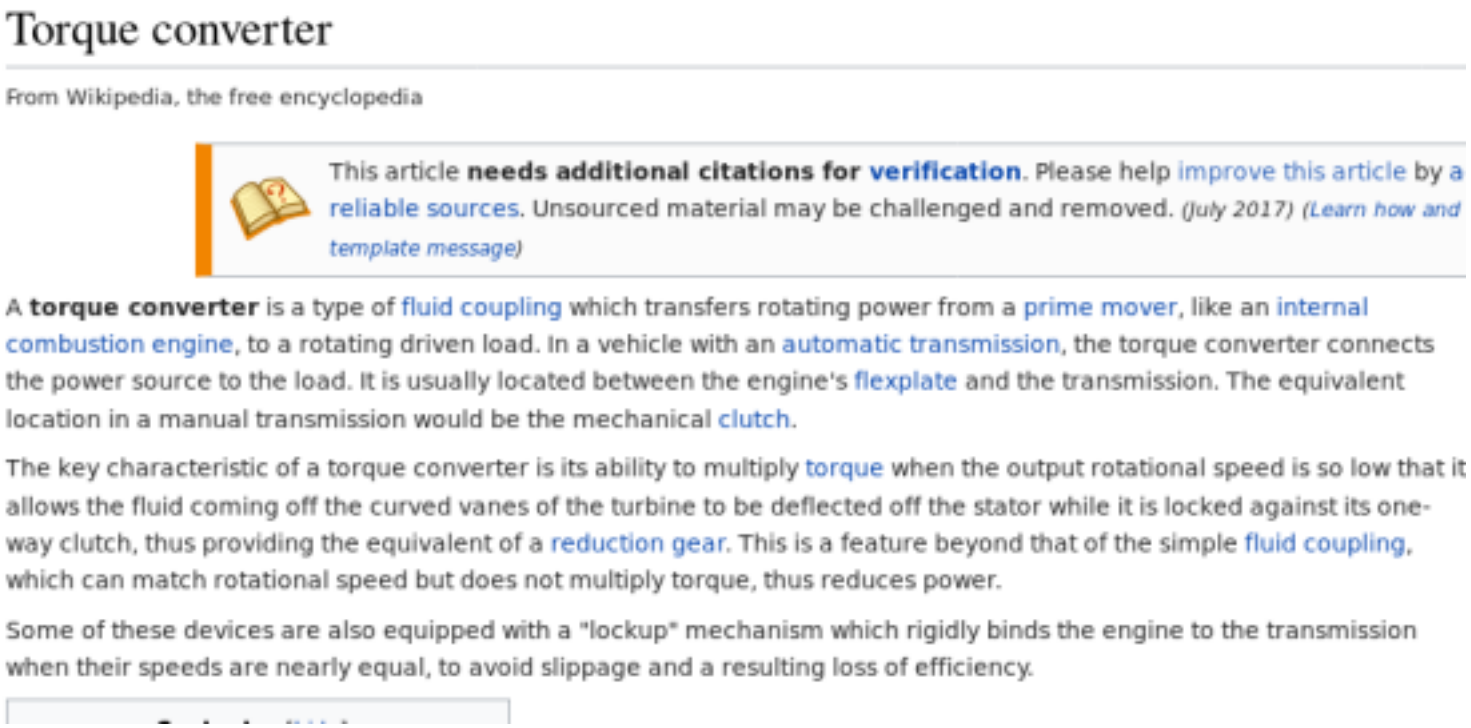
https://en.wikipedia.org/wiki/Torque_converter

Torque Converter Animation

https://www.youtube.com/watch?v=bRcDvCj_JPs - Beautiful Tutorial to understand

https://www.youtube.com/watch?v=EqPQln_gpdc

https://www.youtube.com/watch?v=z5G2zQ_3xTc - Not so much of Information



Torque converter

From Wikipedia, the free encyclopedia

This article **needs additional citations for verification**. Please help improve this article by a reliable sources. Unsourced material may be challenged and removed. *(July 2017) (Learn how and template message)*

A **torque converter** is a type of fluid coupling which transfers rotating power from a prime mover, like an internal combustion engine, to a rotating driven load. In a vehicle with an automatic transmission, the torque converter connects the power source to the load. It is usually located between the engine's flexplate and the transmission. The equivalent location in a manual transmission would be the mechanical clutch.

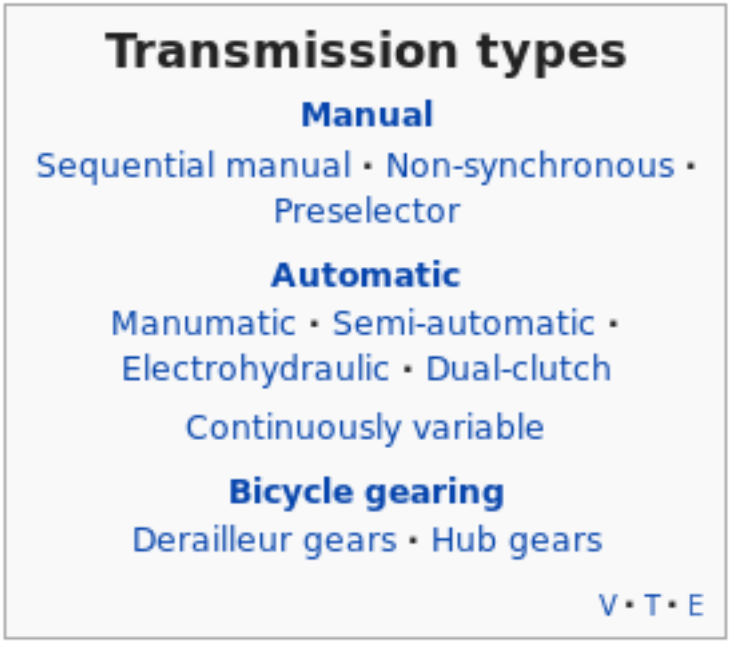
The key characteristic of a torque converter is its ability to multiply torque when the output rotational speed is so low that it allows the fluid coming off the curved vanes of the turbine to be deflected off the stator while it is locked against its one-way clutch, thus providing the equivalent of a reduction gear. This is a feature beyond that of the simple fluid coupling, which can match rotational speed but does not multiply torque, thus reduces power.

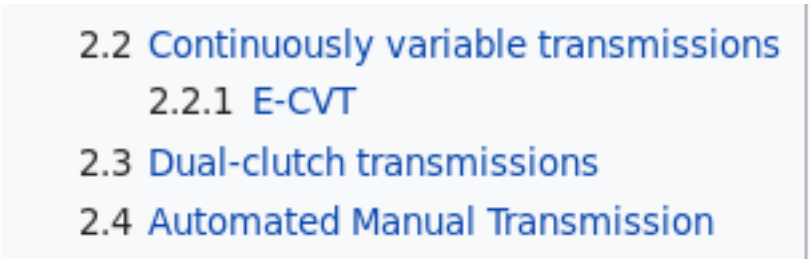
Some of these devices are also equipped with a "lockup" mechanism which rigidly binds the engine to the transmission when their speeds are nearly equal, to avoid slippage and a resulting loss of efficiency.

Transmission

https://en.wikipedia.org/wiki/Transmission_(mechanics)

The most common use is in motor vehicles, where the transmission adapts the output of the internal combustion engine to the drive wheels. Such engines need to operate at a relatively high rotational speed, which is inappropriate for starting, stopping, and slower travel. The transmission reduces the higher engine speed to the slower wheel speed, increasing torque in the process. Transmissions are also used on pedal bicycles, fixed machines, and where different rotational speeds and torques are adapted.





Automatic Transmission

https://en.wikipedia.org/wiki/Automatic_transmission

Continuously Variable Transmission

https://en.wikipedia.org/wiki/Continuously_variable_transmission

Dual Clutch Transmission

https://en.wikipedia.org/wiki/Dual-clutch_transmission

Pros and Cons of Dual Clutch Transmission

https://auto.howstuffworks.com/dual-clutch-transmission2.htm

Driver experience, then, is just one of the many advantages of a DCT. With upshifts taking a mere 8 milliseconds, many
feel that the DCT offers the most dynamic acceleration of any vehicle on the market. It certainly offers smooth
acceleration by eliminating the shift shock that accompanies gearshifts in manual

transmissions and even some
automatics. Best of all, it affords drivers the luxury of choosing whether they prefer to control the shifting or let the
computer do all of the work.

Pros of DCT

Perhaps the most compelling advantage of a DCT is improved fuel economy. Because power flow from the engine to the transmission is not interrupted, fuel efficiency increases dramatically. Some experts say that a six-speed DCT can deliver up to a 10 percent increase in relative fuel efficiency when compared to a conventional five-speed automatic.

Cons of DCT

Many car manufacturers are interested in DCT technology. However, some automakers are wary of the additional costs associated with modifying production lines to accommodate a new type of transmission. This could initially drive up the costs of cars outfitted with DCTs, which might discourage cost-conscious consumers.

Automotive basics [edit]

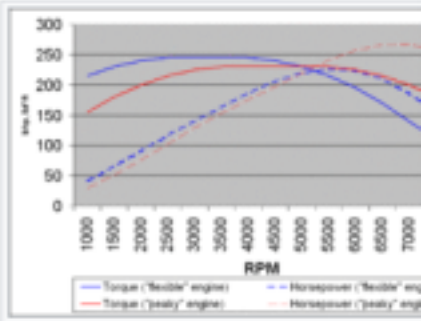
The need for a transmission in an automobile is a consequence of the characteristics of the internal combustion engine. Engines typically operate over a range of 600 to about 7000 rpm (though this varies, and is typically less for diesel engines), while the car's wheels rotate between 0 rpm and around 1800 rpm.

Furthermore, the engine provides its highest torque and power outputs unevenly across the rev range resulting in a torq band and a power band. Often the greatest torque is required when the vehicle is moving from rest or traveling slowly, while maximum power is needed at high speed. Therefore, a system is required that transforms the engine's output so tl it can supply high torque at low speeds, but also operate at highway speeds with the motor still operating within its limits Transmissions perform this transformation.

The dynamics of a car vary with speed: at low speeds, acceleration is limited by the inertia of vehicular gross mass; while at cruising or maximum speeds wind resistance is the dominant barrier.

Many transmissions and gears used in automotive and truck applications are contained in a cast iron case, though more frequently aluminium is used for lower weight especially in cars. There are usually three shafts: a mainshaft, a countershaft, and an idler shaft.

The mainshaft extends outside the case in both directions: the input shaft towards owards the rear axle (on rear wheel drive cars. Front wheel drives generally have nted transversely, the differential being part of the transmission assembly.) The earings, and is split towards the input end. At the point of the split, a pilot bearing ars and clutches ride on the mainshaft, the gears being free to turn relative to the by the clutches.



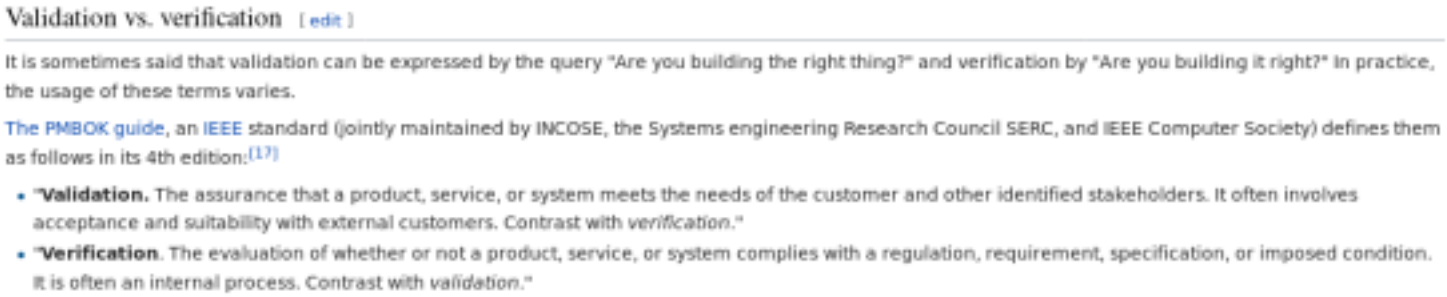
A diagram comparing the power and torque bands of a "torquey" engine versus a "peaky" one

Process

V Model vs Agile

V Model

Validation can be expressed by the query **"Are you building the right thing?"**
Verification by **"Are you building it right?"**



Validation vs. verification [edit]

It is sometimes said that validation can be expressed by the query "Are you building the right thing?" and verification by "Are you building it right?" In practice, the usage of these terms varies.

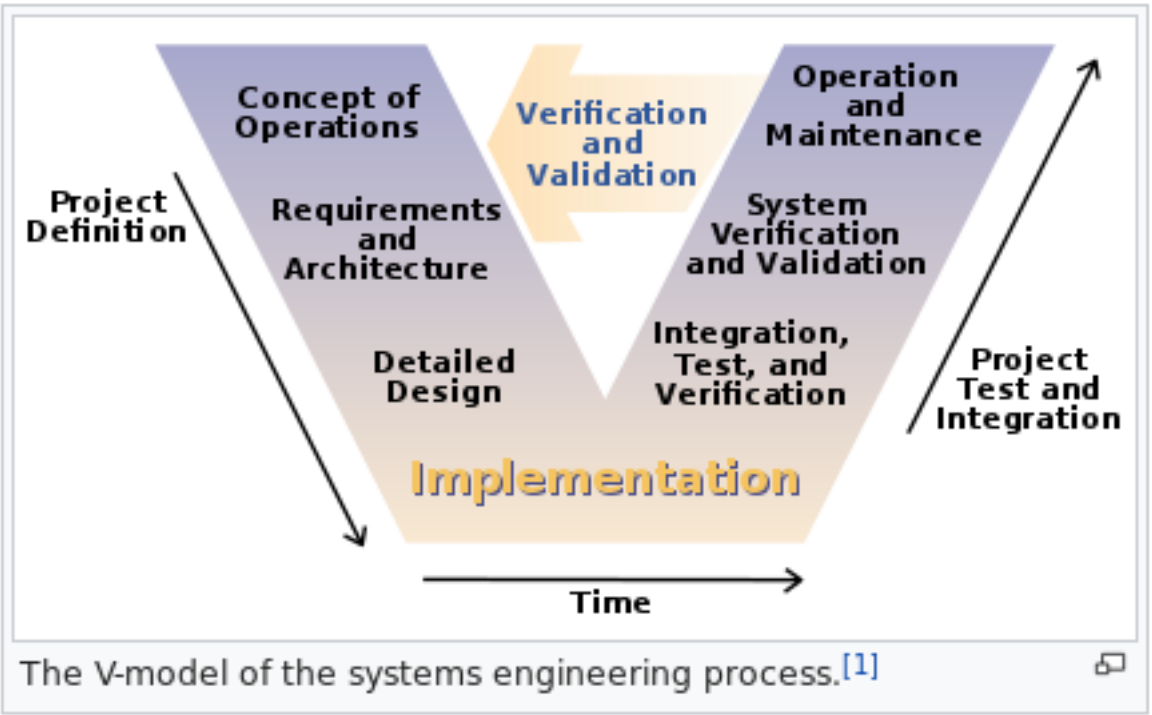
The PMBOK guide, an IEEE standard (jointly maintained by INCOSE, the Systems engineering Research Council SERC, and IEEE Computer Society) defines them as follows in its 4th edition:[17]

- "**Validation.** The assurance that a product, service, or system meets the needs of the customer and other identified stakeholders. It often involves acceptance and suitability with external customers. Contrast with *verification*."
- "**Verification**. The evaluation of whether or not a product, service, or system complies with a regulation, requirement, specification, or imposed condition. It is often an internal process. Contrast with *validation*."

https://en.wikipedia.org/wiki/V-Model

https://en.wikipedia.org/wiki/V-Model_(software_development)

1. Requirement Analysis
2. System Level Design
3. Architecture Design
4. Module Level Design
5. Unit Testing
6. Integration Testing
7. System Testing
8. User Acceptance Testing



The V-model of the systems engineering process.[1]

System testing [edit]

System Tests Plans are developed during System Design Phase. Unlike Unit and Integration Test Plans, System Test Plans are composed by client's business team. System Test ensures that expectations from application developed are met. The whole application is tested for its functionality, interdependency and communication. System Testing verifies that functional and non-functional requirements have been met. Load and performance testing, stress testing, regression testing, etc., are subsets of system testing.

regression testing, etc., are subsets of system testing.

User acceptance testing [edit]

User Acceptance Test (UAT) Plans are developed during the Requirements Analysis phase. Test Plans are composed by business users. UAT is performed in a user environment that resembles the production environment, using realistic data. UAT verifies that delivered system meets user's requirement and system is ready for use in real time.

Agile Model

Agile software development values [edit]

Based on their combined experience of developing software and helping
value:[4]

- ***Individuals and Interactions*** *over processes and tools*
- ***Working Software*** *over comprehensive documentation*
- ***Customer Collaboration*** *over contract negotiation*
- ***Responding to Change*** *over following a plan*

That is, while there is value in the items on the right, they value the items

12 Principles in Agile

Agile software development principles [edit]

The *Manifesto for Agile Software Development* is based on twelve principles:[21]

1. Customer satisfaction by early and continuous delivery of valuable software
2. Welcome changing requirements, even in late development
3. Working software is delivered frequently (weeks rather than months)
4. Close, daily cooperation between business people and developers
5. Projects are built around motivated individuals, who should be trusted
6. Face-to-face conversation is the best form of communication (co-location)
7. Working software is the primary measure of progress
8. Sustainable development, able to maintain a constant pace
9. Continuous attention to technical excellence and good design
10. Simplicity—the art of maximizing the amount of work not done—is essential
11. Best architectures, requirements, and designs emerge from self-organizing teams
12. Regularly, the team reflects on how to become more effective, and adjusts accordingly

Agile software development describes an approach to software development under which requirements and solutions evolve through the collaborative effort of self-organizing and cross-functional teams and their customer(s)/end user(s).[1] It advocates adaptive planning, evolutionary development, early delivery, and continual improvement, and it encourages rapid and flexible response to change.[2]

The term *agile* (sometimes written *Agile*)[3] was popularized, in this context, by the *Manifesto for Agile Software Development*.[4] The values and principles espoused in this manifesto were derived from and underpin a broad range of software development frameworks, including Scrum and Kanban.[5][6]

There is significant anecdotal evidence that adopting agile practices and values improves the agility of software professionals, teams and organizations; however, some empirical studies have found no scientific evidence.[7][8]

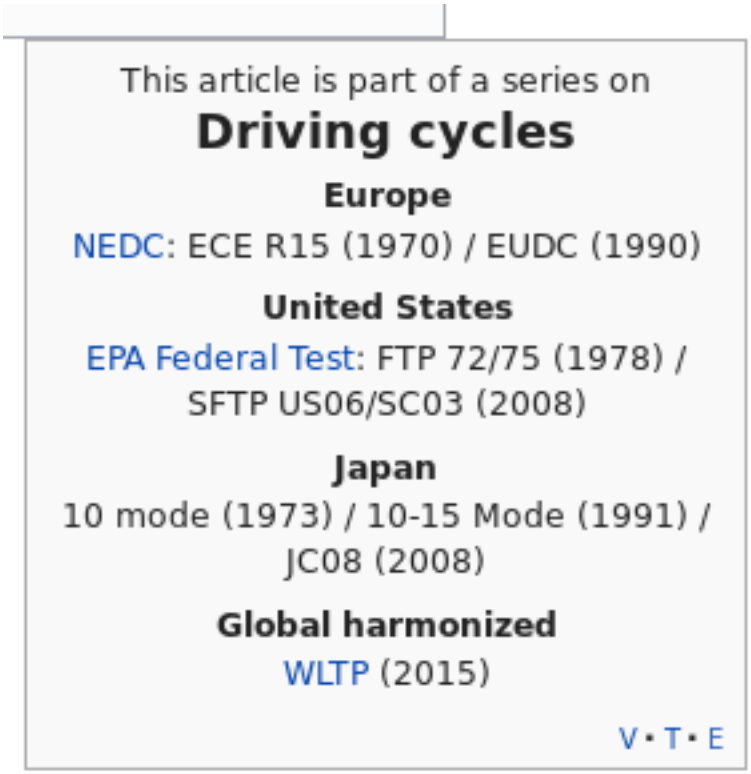
Home grounds of different development methods

Value-driven methods	Plan-driven methods	Formal methods
Low criticality	High criticality	Extreme criticality
Senior developers	Junior developers(?)	Senior developers
Requirements change often	Requirements do not change often	Limited requirements, limited features see Wirth's law*[clarification needed]*
Small number of developers	Large number of developers	Requirements that can be modeled
Culture that responds to change	Culture that demands order	Extreme quality

Standards

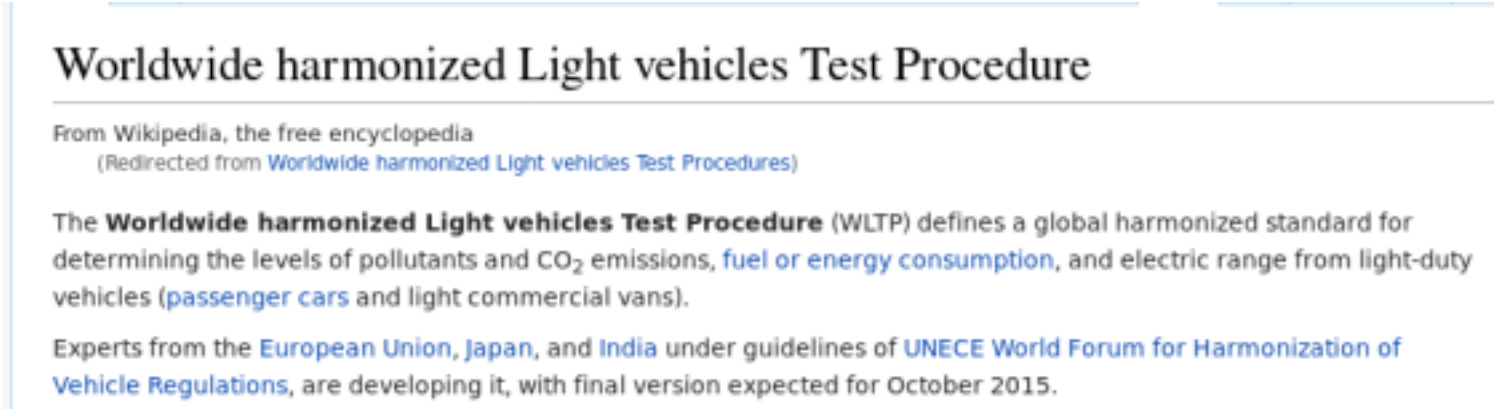
Driving Cycles

https://en.wikipedia.org/wiki/Driving_cycle



All Driving Cycles with Graphs

http://www.car-engineer.com/the-different-driving-cycles/



Worldwide harmonized Light vehicles Test Procedure

From Wikipedia, the free encyclopedia
(Redirected from Worldwide harmonized Light vehicles Test Procedures)

The **Worldwide harmonized Light vehicles Test Procedure** (WLTP) defines a global harmonized standard for determining the levels of pollutants and CO_2 emissions, fuel or energy consumption, and electric range from light-duty vehicles (passenger cars and light commercial vans).

Experts from the European Union, Japan, and India under guidelines of UNECE World Forum for Harmonization of Vehicle Regulations, are developing it, with final version expected for October 2015.

WLTP3>135

WLTC Class 3 test cycle

	Low	Medium	High	Extra High	Total
Duration, s	589	433	455	323	1800
Stop duration, s	150	49	31	8	235
Distance, m	3095	4756	7162	8254	23266
% of stops	26.5%	11.1%	6.8%	2.2%	13.4%
Maximum speed, km/h	56.5	76.6	97.4	131.3	
Average speed without stops, km/h	25.3	44.5	60.7	94.0	53.5
Average speed with stops, km/h	18.9	39.4	56.5	91.7	46.5
Minimum acceleration, m/s^2	-1.5	-1.5	-1.5	-1.44	
Maximum acceleration, m/s^2	1.611	1.611	1.666	1.055	

AUTOSAR

https://en.wikipedia.org/wiki/AUTOSAR

https://www.autosar.org/standards/classic-platform/

AUTOSAR

From Wikipedia, the free encyclopedia

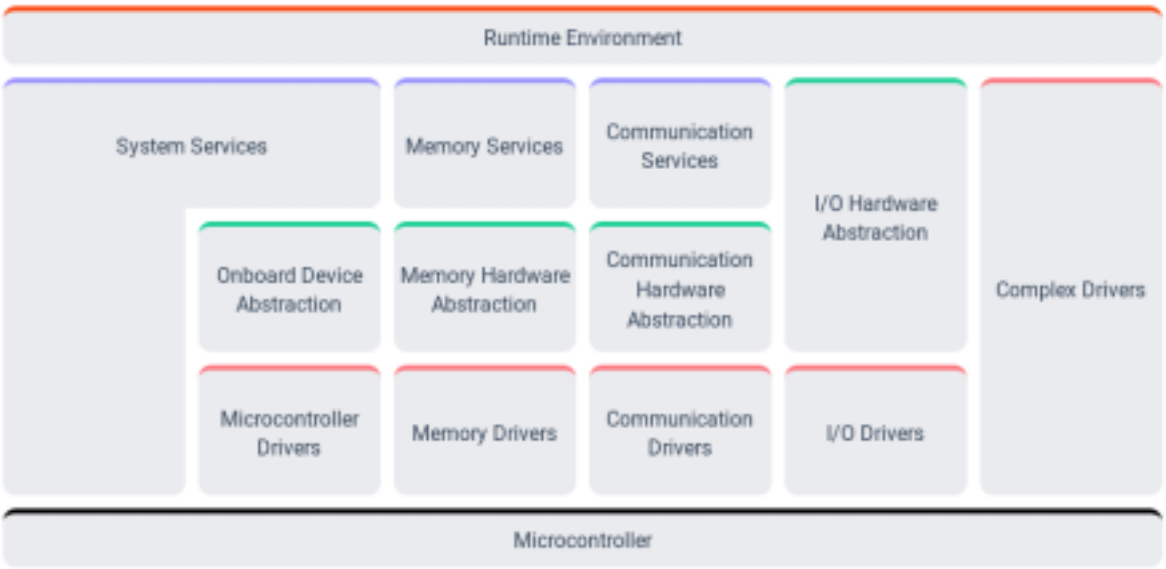
AUTOSAR (AUTomotive Open System ARchitecture) is a worldwide development partnership of automotive interested parties founded in 2003. It pursues the objective of creating and establishing an open and standardized software architecture for automotive electronic control units (ECUs). Goals include the scalability to different vehicle and platform variants, transferability of software, the consideration of availability and safety requirements, a collaboration between various partners, sustainable utilization of natural resources, and maintainability throughout the whole "Product Life Cycle".[1][2]

Software architecture [edit]

AUTOSAR uses a three-layered architecture:

- Basic Software: standardized software modules (mostly) without any functional job itself that offers services necessary to run the functional part of the upper software layer.[9]
- Runtime environment(RTE): Middleware which abstracts from the network topology for the inter- and intra-ECU information exchange between the application software components and between the Basic Software and the applications.[10]
- Application Layer: application software components that interact with the runtime environment.[11]

SOFTWARE ARCHITECTURE



CLASSIC PLATFORM

ABOUT

The AUTOSAR Classic Platform architecture distinguishes on the highest abstraction level between three software layers which run on a microcontroller: application, runtime environment (RTE) and basic software (BSW).

- The application software layer is mostly hardware independent.
- Communication between software components and access to BSW via RTE.
- The RTE represents the full interface for applications.
- The BSW is divided in three major layers and complex drivers:
  - Services, ECU (Electronic Control Unit) abstraction and microcontroller abstraction.
- Services are divided furthermore into functional groups representing the infrastructure for system, memory and communication services.

ISTQB

https://www.istqb.org/certification-path-root/foundation-level-2018/foundation-level-2018-content.html

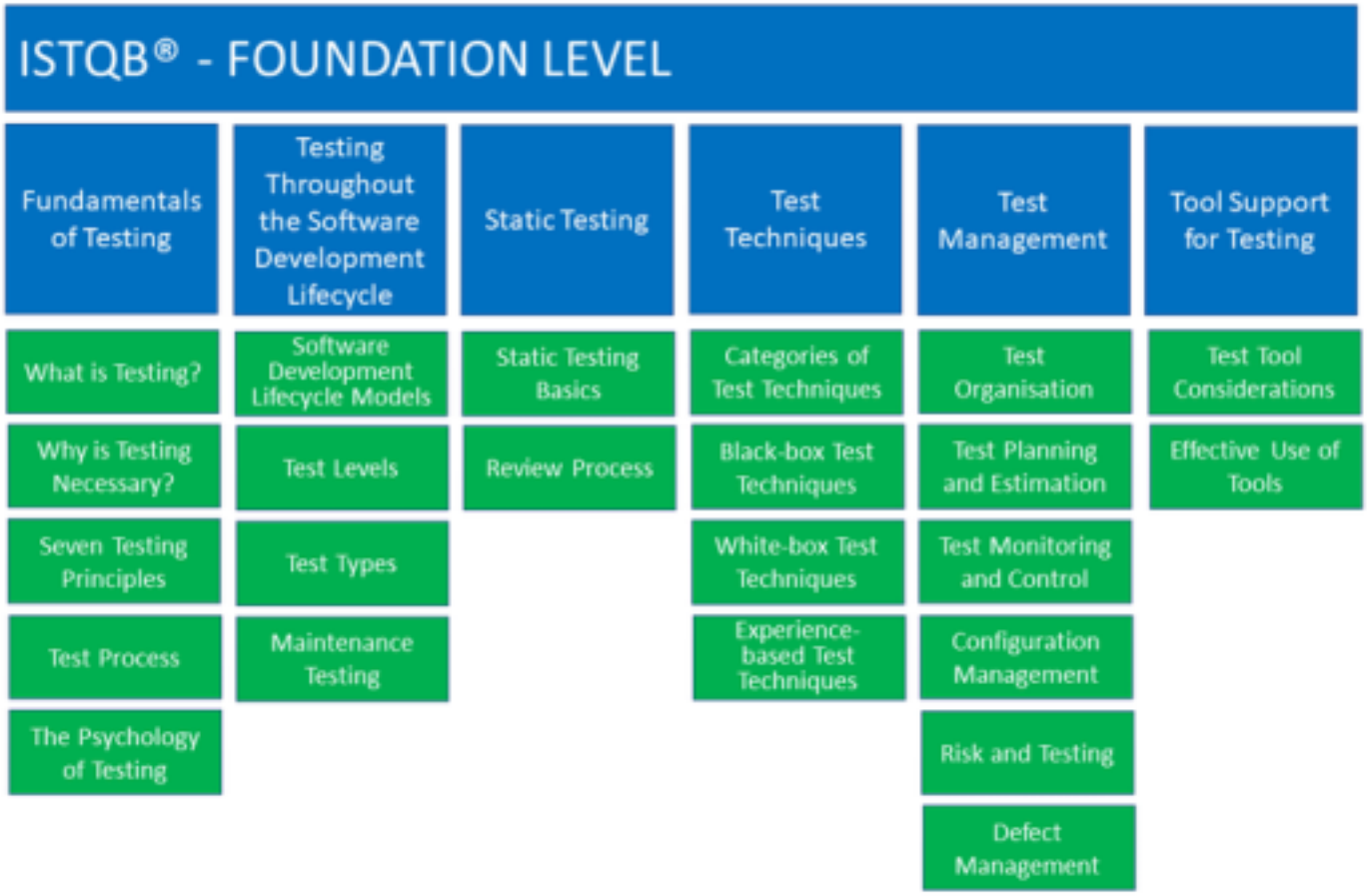
https://www.istqb.org/

https://en.wikipedia.org/wiki/International_Software_Testing_Qualifications_Board

International Software Testing Qualifications Board

The International Software Testing Qualifications Board is a software testing qualification certification organisation that operates internationally. Founded in Edinburgh in November 2002, ISTQB is a non-profit association legally registered in Belgium.

W More at Wikipedia


ISTQB® - FOUNDATION LEVEL
Fundamentals of Testing
What is Testing?
Why is Testing Necessary?
Seven Testing Principles
Test Process
The Psychology of Testing
Testing Throughout the Software Development Lifecycle
Software Development Lifecycle Models
Test Levels
Test Types
Maintenance Testing
Static Testing
Static Testing Basics
Review Process
Test Techniques
Categories of Test Techniques
Black-box Test Techniques
White-box Test Techniques
Experience-based Test Techniques
Test Management
Test Organisation
Test Planning and Estimation
Test Monitoring and Control
Configuration Management
Risk and Testing
Defect Management
Tool Support for Testing
Test Tool Considerations
Effective Use of Tools

Objectives

1.1.1 Typical Objectives of Testing

For any given project, the objectives of testing may include:

- To evaluate work products such as requirements, user stories, design, and code
- To verify whether all specified requirements have been fulfilled
- To validate whether the test object is complete and works as the users and other stakeholders expect
- To build confidence in the level of quality of the test object
- To prevent defects
- To find failures and defects
- To provide sufficient information to stakeholders to allow them to make informed decisions, especially regarding the level of quality of the test object
- To reduce the level of risk of inadequate software quality (e.g., previously undetected failures occurring in operation)
- To comply with contractual, legal, or regulatory requirements or standards, and/or to verify the test object's compliance with such requirements or standards

1.2.3 Errors, Defects, and Failures

A person can make an error (mistake), which can lead to the introduction of a defect (fault or bug) in the software code or in some other related work product. An error that leads to the introduction of a defect in one work product can trigger an error that leads to the introduction of a defect in a related work product. For example, a requirements elicitation error can lead to a requirements defect, which then results in a programming error that leads to a defect in the code.

If a defect in the code is executed, this may cause a failure, but not necessarily in all circumstances. For example, some defects require very specific inputs or preconditions to trigger a failure, which may occur rarely or never.

Errors may occur for many reasons, such as:

- Time pressure
- Human fallibility
- Inexperienced or insufficiently skilled project participants
- Miscommunication between project participants, including miscommunication about requirements and design
- Complexity of the code, design, architecture, the underlying problem to be solved, and/or the technologies used
- Misunderstandings about intra-system and inter-system interfaces, especially when such intra-system and inter-system interactions are large in number
- New, unfamiliar technologies

7 Principles of Testing

1.3 Seven Testing Principles

A number of testing principles have been suggested over the past 50 years and offer general guidelines common for all testing.

1. Testing shows the presence of defects, not their absence

Testing can show that defects are present, but cannot prove that there are no defects. Testing reduces the probability of undiscovered defects remaining in the software but, even if no defects are found, testing is not a proof of correctness.

2. Exhaustive testing is impossible

Testing everything (all combinations of inputs and preconditions) is not feasible except for trivial cases. Rather than attempting to test exhaustively, risk analysis, test techniques, and priorities should be used to focus test efforts.

3. Early testing saves time and money

To find defects early, both static and dynamic test activities should be started as early as possible in the software development lifecycle. Early testing is sometimes referred to as *shift left*. Testing early in the software development lifecycle helps reduce or eliminate costly changes (see section 3.1).

4. Defects cluster together

A small number of modules usually contains most of the defects discovered during pre-release testing, or is responsible for most of the operational failures. Predicted defect clusters, and the actual observed defect clusters in test or operation, are an important input into a risk analysis used to focus the test effort (as mentioned in principle 2).

5. Beware of the pesticide paradox

If the same tests are repeated over and over again, eventually these tests no longer find any new defects. To detect new defects, existing tests and test data may need changing, and new tests may need to be written. (Tests are no longer effective at finding defects, just as pesticides are no longer effective at killing insects after a while.) In some cases, such as automated regression testing, the pesticide paradox has a beneficial outcome, which is the relatively low number of regression defects.

6. Testing is context dependent

Testing is done differently in different contexts. For example, safety-critical industrial control software is tested differently from an e-commerce mobile app. As another example, testing in an Agile project is done differently than testing in a sequential lifecycle project (see section 2.1).

7. Absence-of-errors is a fallacy

Some organizations expect that testers can run all possible tests and find all possible defects, but principles 2 and 1, respectively, tell us that this is impossible. Further, it is a fallacy (i.e., a mistaken belief) to expect that *just* finding and fixing a large number of defects will ensure the success of a system. For example, thoroughly testing all specified requirements and fixing all defects found could still produce a system that is difficult to use, that does not fulfill the users' needs and expectations, or that is inferior compared to other competing systems.

Types

2.2 Test Levels

Test levels are groups of test activities that are organized and managed together. Each test level is an instance of the test process, consisting of the activities described in section 1.4, performed in relation to software at a given level of development, from individual units or components to complete systems or, where applicable, systems of systems. Test levels are related to other activities within the software development lifecycle. The test levels used in this syllabus are:

- Component testing
- Integration testing
- System testing
- Acceptance testing

MISRA

https://en.wikipedia.org/wiki/Motor_Industry_Software_Reliability_Association

Motor Industry Software Reliability Association

Motor Industry Software Reliability Association is an organization that produces guidelines for the software developed for electronic components used in the automotive industry. It is a collaboration between vehicle manufacturers, component suppliers and engineering consultancies.

W More at Wikipedia

ISO26262

https://en.wikipedia.org/wiki/ISO_26262

Functional Safety Requirements

> **Severity Classifications (S):**
>
> **S0** No Injuries
>
> **S1** Light to moderate injuries
>
> **S2** Severe to life-threatening (survival probable) injuries
>
> **S3** Life-threatening (survival uncertain) to fatal injuries

Exposure Classifications (E):

E0 Incredibly unlikely

E1 Very low probability (injury could happen only in rare operating conditions)

E2 Low probability

E3 Medium probability

E4 High probability (injury could happen under most operating conditions)

Controllability Classifications (C):

C0 Controllable in general

C1 Simply controllable

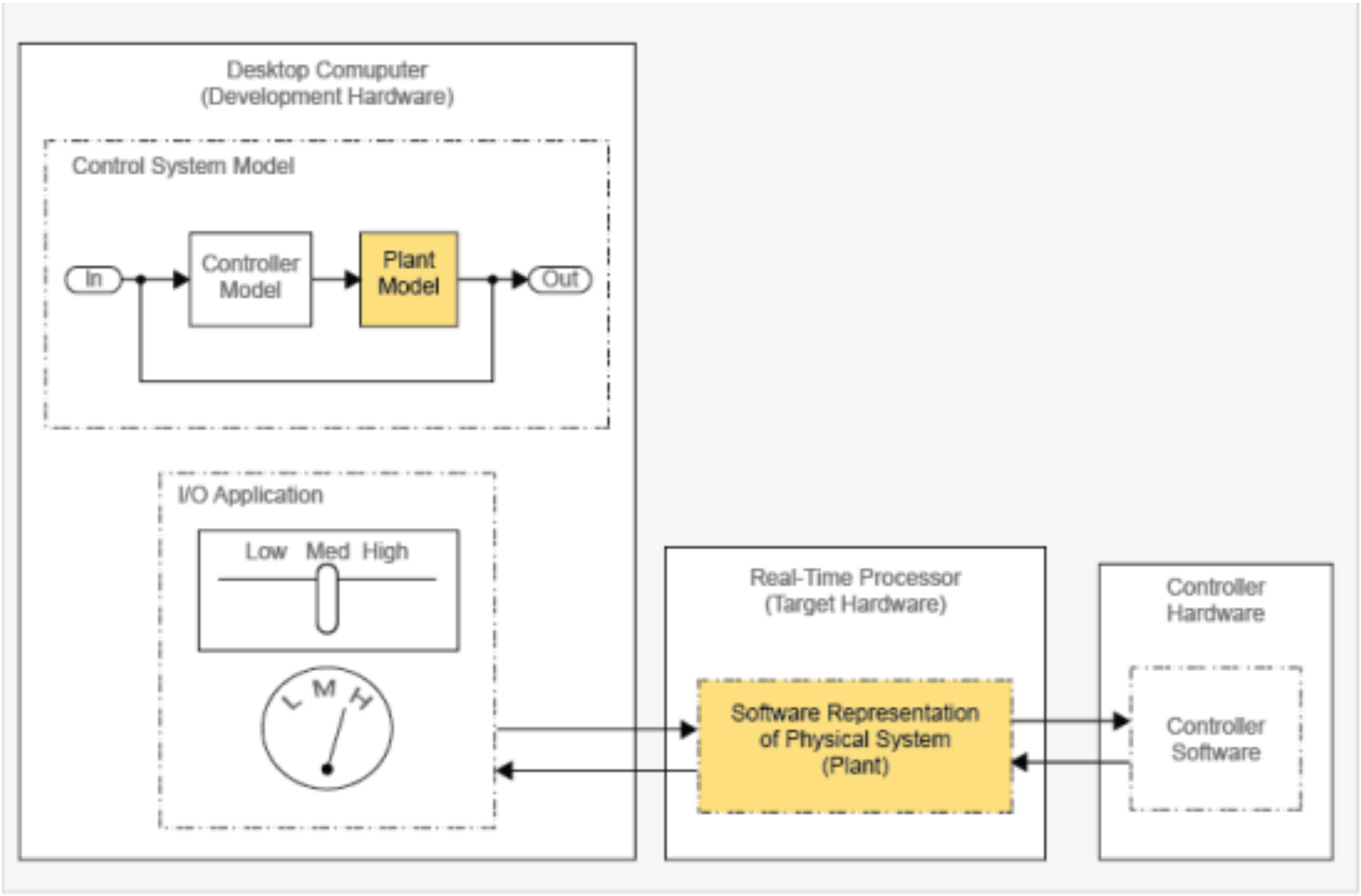
C2 Normally controllable (most drivers could act to prevent injury)

C3 Difficult to control or uncontrollable

Testing Methodology

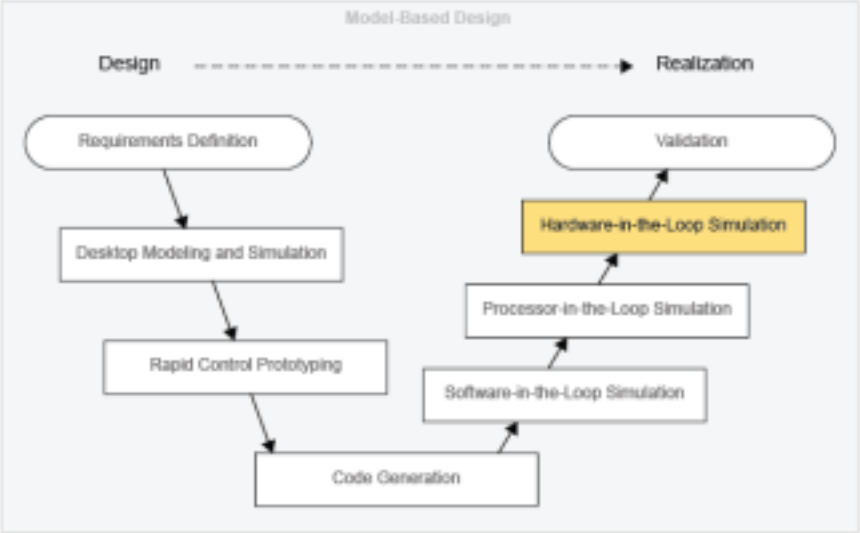
Hardware In Loop

https://en.wikipedia.org/wiki/Hardware-in-the-loop_simulation



Why Perform Hardware-In-The-Loop Simulation?

Use HIL simulation to test the design of your controller when you are performing Model-Based Design (MBD). The figure shows where HIL simulation fits into the MBD design-to-realization workflow.



https://in.mathworks.com/help/physmod/simscape/ug/what-is-hardware-in-the-loop-simulation.html

How HIL works [edit]

A HIL simulation must include electrical emulation of sensors and actuators. These electrical emulations act as the interface between the plant simulation and the embedded system under test. The value of each electrically emulated sensor is controlled by the plant simulation and is read by the embedded system under test (feedback). Likewise, the embedded system under test implements its control algorithms by outputting actuator control signals. Changes in the control signals result in changes to variable values in the plant simulation.

For example, a HIL simulation platform for the development of automotive anti-lock braking systems may have mathematical representations for each of the following subsystems in the plant simulation:[1]

- Vehicle dynamics, such as suspension, wheels, tires, roll, pitch and yaw;
- Dynamics of the brake system's hydraulic components;
- Road characteristics.

Based on
1. Cost
2. Duration
3. Safety
4. Feasibility

To Simulate extremely unlikely events like Drastic Climatic Conditions, EarthQuakes, Implausible Inputs are possible
@ HIL

Uses [edit]

In many cases, the most effective way to develop an embedded system is to connect the embedded system to the real plant. In other cases, HIL simulation is more efficient. The metric of development and test efficiency is typically a formula that includes the following factors: 1. Cost 2. Duration 3. Safety 4. Feasibility

The cost of the approach should be a measure of the cost of all tools and effort. The duration of development and testing affects the time-to-market for a planned product. Safety factor and development duration are typically equated to a cost measure. Specific conditions that warrant the use of HIL simulation include the following:

- Enhancing the quality of testing
- Tight development schedules
- High-burden-rate plant
- Early process human factor development

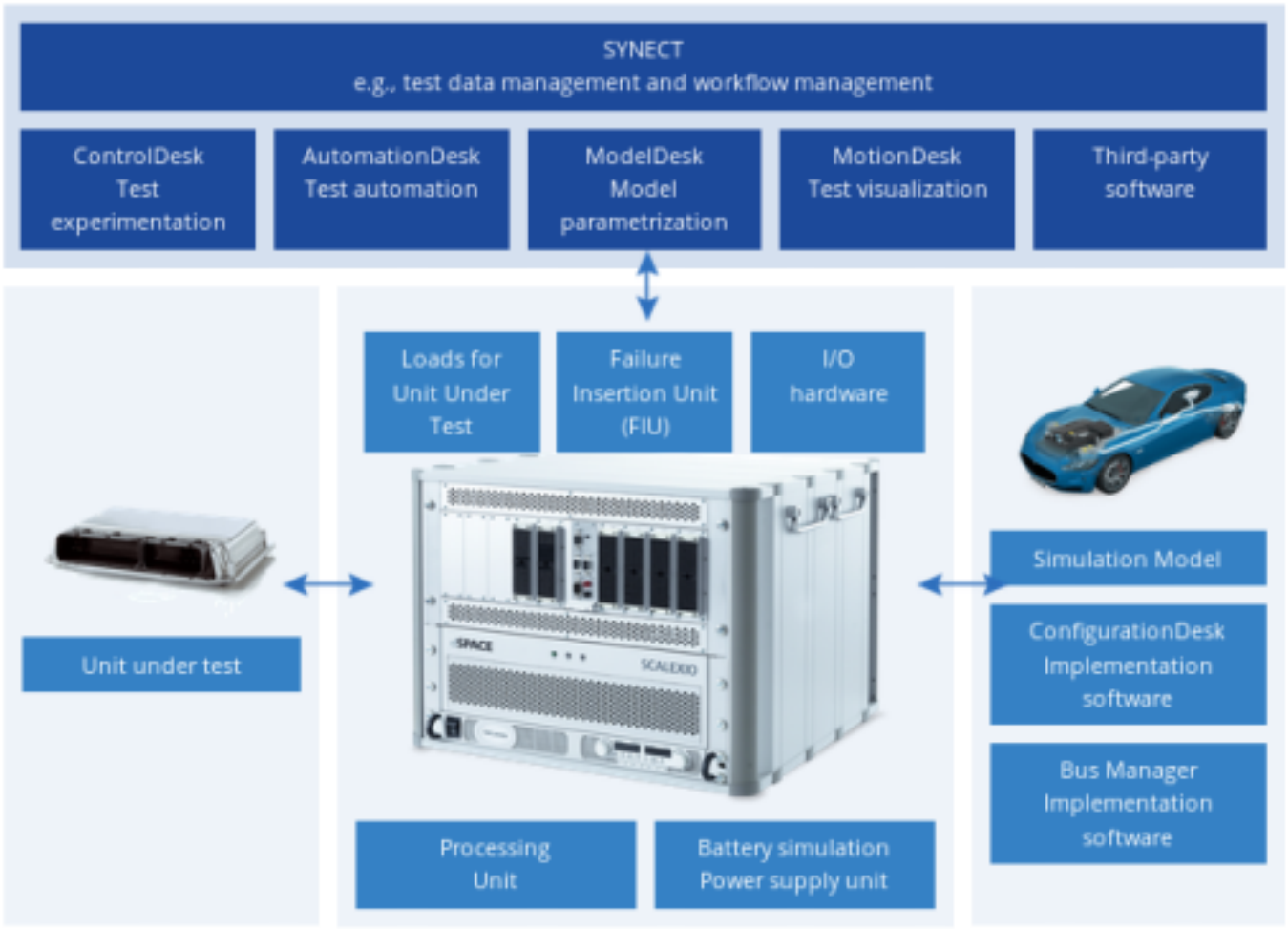
Automotive systems [edit]

In context of automotive applications "Hardware-in-the-loop simulation systems provide such a virtual vehicle for systems validation and verification."[2] Since in-vehicle driving tests for evaluating performance and diagnostic functionalities of Engine Management Systems are often time-consuming, expensive and not reproducible, HIL simulators allow developers to validate new hardware and software automotive solutions, respecting quality requirements and time-to-market restrictions. In a typical HIL Simulator, a dedicated real-time processor executes mathematical models which emulate engine dynamics. In addition, an I/O unit allows the connection of vehicle sensors and actuators (which usually present high degree of non-linearity). Finally, the Electronic Control Unit (ECU) under test is connected to the system and stimulated by a set of vehicle maneuvers executed by the simulator. At this point, HIL simulation also offers a high degree of repeatability during testing phase.

In the literature, several HIL specific applications are reported and simplified HIL simulators were built according to some specific purpose.[1][3][4] When testing a new ECU software release for example, experiments can be performed in open loop and therefore several engine dynamic models are no longer required. The strategy is restricted to the analysis of ECU outputs when excited by controlled inputs. In this case, a Micro HIL system (MHIL) offers a simpler and more economic solution.[5] Since complexity of models processing is dumped, a full-size HIL system is reduced into a portable device composed of a signal generator, an I/O board, and a console containing the actuators (external loads) to be connected to the ECU.

dSPACE HIL

https://www.dspace.com/en/pub/home/products/systems/ecutest/hil_system.cfm



A typical HIL test system comprises components such as:

- A processing unit for computing large and complex simulation models and for connecting the real-time model with the relevant I/O. The dSPACE SCALEXIO processing hardware therefore provides multicore and multiprocessor support as well as communication with the I/O via IOCNET for handling a high number of I/O and bus channels in real-time.
- Battery simulation components for providing the power supply to generate currents and voltages for the ECU for manual and automated tests. dSPACE offers various boards for controlling the battery simulation power supply, e.g., the DS2907 Battery Simulation Controller, the DS2642 FIU & Power Switch Board, and the DS2680 I/O Unit.
- I/O interfaces for connecting the ECU with the test system. For different test scenarios, dSPACE offers various boards with dedicated or freely configurable I/O interfaces, such as the SCALEXIO MultiCompact, HighFlex, and I/O boards.
- Auxiliary components, such as load boards, and the possibility to perform electrical fault simulation for specific application scenarios. For this, dSPACE offers the Electronic Load Modules and the DS2642 FIU & Power Switch Board.
- A simulation of the bus and network communication to test the communication between two or more EUCs. Network participants that are not available yet are simulated as well.
- One or more ECUs containing new functions or ECU software to be tested. They are connected to the test system and typically called device under test (DUT).
- The software for configuring and automating the HIL test. It runs on a PC, as does the software for parameterizing the simulation model and visualizing the simulation. Data management software can be used to handle and manage test data, such as models, signals, parameters, tests, and test results.

History of dSPACE products [edit]

- 1988: First real-time development system for control technology/mechatronics, based on a digital signal processor
- 1989: First hardware-in-the-loop (HIL) simulator is shipped
- 1990: First real-time development system with a floating-point processor is shipped
- 1992: RTI, first real-time system connected to MATLAB/Simulink
- 1994: First multiprocessor hardware for real-time development systems
- 1995: First turnkey (HIL) simulator for an ABS/ESP test bench
- 1999: MicroAutoBox, a complete prototyping system for in-vehicle use
- 1999: TargetLink, the first production code generator for ECUs based on MATLAB/Simulink
- 2003: CalDesk, a component of the dSPACE calibration system
- 2005: RapidPro, a modular system for signal conditioning and power stages
- 2005: Automotive Simulation Models (ASMs), real-time automotive simulation models based on MATLAB/Simulink
- 2007: SystemDesk,[12] tool for developing complex ECU software architectures based on the AUTOSAR concept
- 2010: MicroAutoBox II, second generation of the vehicle-capable prototyping systems
- 2011: SCALEXIO, the new hardware-in-the-loop system, including new ConfigurationDesk configuration software
- 2012: VEOS®,[13] PC-based simulation platform for early validation of ECU software
- 2015: MicroLabBox: Compact prototyping unit for the laboratory[14]

Model In Loop

Software In Loop

Tools & Technology

logi.cals

logi.CAD logi.DOC Additional services About us With us News Support

The programming platform for industrial controllers

logi.CAD 3 is the engineering software for creating controller applications for industrial automation. Systems of all kinds can be programmed in accordance with the industry norm IEC 61131-3, from the microcontroller to various OEM platforms and multi-core industrial PCs.

logi.CAD 3, together with the runtime system logi.RTS, enables the implementation of cost-effective, powerful controller platforms (custom PLCs) on diverse hardware systems. This results in solutions that are precisely configured to meet machine and system manufacturer requirements.

MATLAB

m Script

```
function max = mymax(n1, n2, n3, n4, n5)

%This function calculates the maximum of the
% five numbers given as input
max =  n1;
if(n2 > max)
   max = n2;
end
if(n3 > max)
   max = n3;
end
if(n4 > max)
   max = n4;
end
if(n5 > max)
   max = n5;
end
```

Use of Integrators/Differentiators in a Model

When you want the value to slowly increase in Time - Integrator
When you want the value to slowly decrease in Time - Differentiator

Different MATLAB Versions

MATLAB 8.2	R2013b	30	1.7.0_11	2013	September 6, 2013[70]	New table data type[71]
MATLAB 8.3	R2014a	31	1.7.0_11	2014	March 7, 2014[72]	Simplified compiler setup for building MEX-files; USB Webcams support in core MATLAB; number of local workers no longer limited to 12 with Parallel Computing Toolbox
MATLAB 8.4	R2014b	32	1.7.0_11		October 3, 2014	New class-based graphics engine (a.k.a. HG2);[73] tabbing function in GUI;[74] improved user toolbox packaging and help files;[75] new objects for time-date manipulations;[76] Git-Subversion integration in IDE;[77] big data abilities with MapReduce (scalable to Hadoop);[78] new `py` package for using Python from inside MATLAB,[79] new engine interface to call MATLAB from Python;[80] several new and improved functions: `webread` (RESTful web services with JSON/XML support), `tcpclient` (socket-based connections), `histcounts`, `histogram`, `animatedline`, and others
MATLAB 8.5	R2015a	33	1.7.0_60	2015	March 5, 2015	Last release supporting Windows XP and Windows Vista
MATLAB 8.5	R2015aSP1		1.7.0_60		October 14, 2015	
MATLAB 8.6	R2015b	34	1.7.0_60		September 3, 2015	New MATLAB execution engine (a.k.a. LXE);[81] `graph` and `digraph` classes to work with graphs and networks;[82] MinGW-w64 as supported compiler on Windows;[83] Last version with 32-bit support
MATLAB 9.0	R2016a	35	1.7.0_60	2016	March 3, 2016	Live Scripts: interactive documents that combine text, code, and output (in the style of Literate programming);[84] App Designer: a new development environment for building apps (with new kind of UI figures, axes, and components);[85] pause execution of running programs using a Pause Button

---

what is MAAB?

MAAB-MATLAB Automotive Advisory Board which provide Modeling Guidelines

Tell Some Example Guidelines for MAAB?
Some Rules

difference between mux,demux,bus creator and bus selector?

mux cant have different data typed signals as input while bus can have different data type signals.
mux works with 2^n inputs and n outputs while demux would do the reverse with n inputs and 2^n outputs
bus creator is used to create a bus while bus selector is used to select the respective signal based on the index

what is a vector in MATLAB?how can a vector be represented?
any vector can be represented by means of matrix representation in two or more rows?

yes.vectors can be represented in two or more rows in matlab.

what are solvers?

solvers are algorithms used to find the system output at every time instant.
different types of solvers exist of which different type of algorithms are used like odeman-prince,bogacki-shampine etc.
Two types of solvers exist
1.Fixed step size solver where the sample time is fixed in terms of 10ms,20ms etc.,
2.Variable step size solver where the sample time is dynamically varied to get a more precise system response.
  The underlying logic would be like it would predominantly

captures the sample whenever there is a change in the output.

Difference between gain block and product block?

gain block is used instead of product block because in product block,matrix size is to be determined and the input is to be given.but in gain unanimously all the elements in a matrix be multiplied by that factor.Answer is wrong both does the same operation when the multiplication parameter in product block is selected as element wise.But when selected as matrix wise in product block then the matrix multiplication is done

what is Relay Block?

Relay block is the replacement for hysteresis block.The output will follow the input until it is above a threshold.
Say if A is the input,
then if A>Upper Threhold then output =1
then if Lower Threshold <A<Upper Threshold then output =1
then if A<Lower Threhold then output =0
then if Lower Threshold<A<Upper Threshold then output =0

Octave

https://en.wikipedia.org/wiki/GNU_Octave

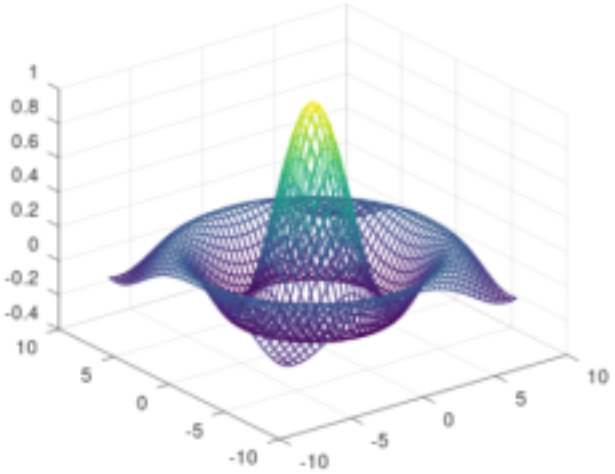
Real Alternative to MATLAB Programming Platform.
Provides Bidirectional Semantical/Syntactical Compatibility with MATLAB

GNU Octave

From Wikipedia, the free encyclopedia

GNU Octave is software featuring a high-level programming language, primarily intended for numerical computations. Octave helps in solving linear and nonlinear problems numerically, and for performing other numerical experiments using a language that is mostly compatible with MATLAB. It may also be used as a batch-oriented language. Since it is part of the GNU Project, it is free software under the terms of the GNU General Public License.

Octave is one of the major free alternatives to MATLAB, others being Scilab and FreeMat.[5][6][7][8] Scilab, however, puts less emphasis on (bidirectional) syntactic compatibility with MATLAB than Octave does.[5][9][10]



SciLab & Xcos

https://en.wikipedia.org/wiki/Scilab

Scilab is a free and open-source, cross-platform numerical computational package and a high-level, numerically oriented programming language. It can be used for signal processing, statistical analysis, image enhancement, fluid dynamics simulations, numerical optimization, and modeling, simulation of explicit and implicit dynamical systems and (if the corresponding toolbox is installed) symbolic manipulations.

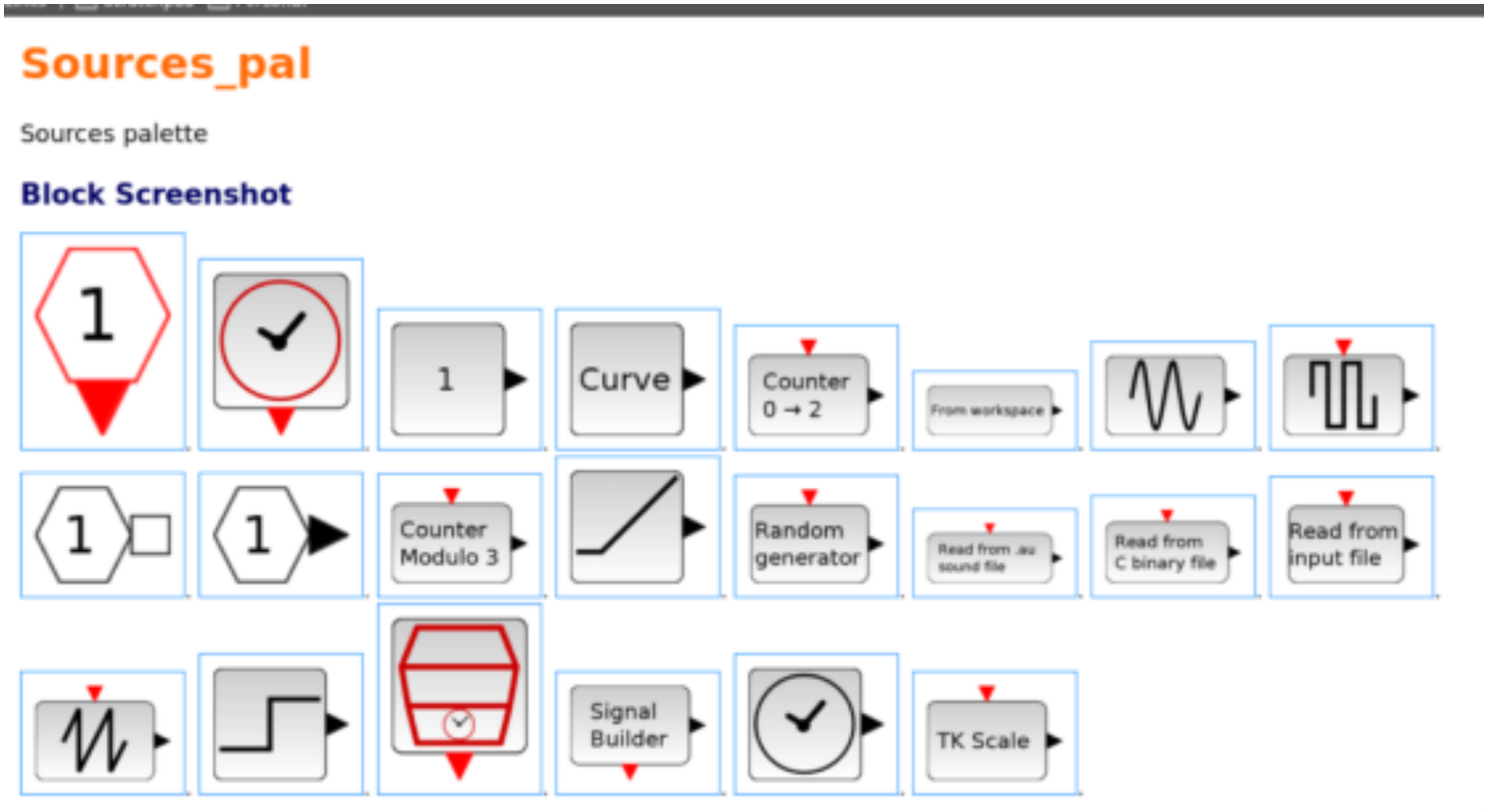
Scilab is one of the two major open-source alternatives to MATLAB, the other one being GNU Octave.[1][2][3][4] Scilab puts less emphasis on syntactic compatibility with MATLAB than Octave does, [1][5][6] but it is similar enough that some authors suggest that it is easy to transfer skills between the two systems.[7]

numerical optimization.[8][9][10]

Scilab also includes a free package called Xcos (a fork of Scicos based on Modelica language) for modeling and simulation of explicit and implicit dynamical systems, including both continuous and discrete sub-systems. Xcos is the open source equivalent to Simulink from the MathWorks.

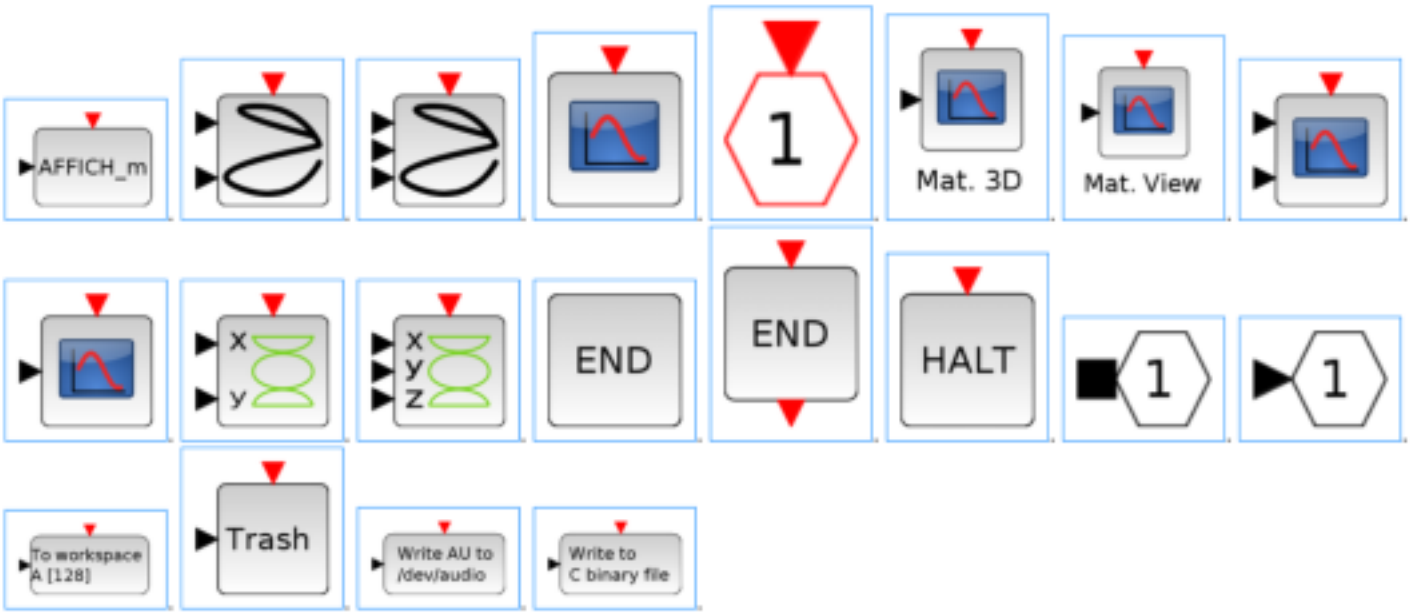
As the syntax of Scilab is similar to MATLAB, Scilab includes a source code translator for assisting the conversion of code from MATLAB to Scilab. Scilab is available free of cost under an open source license. Due to the open source nature of the software, some user contributions have been integrated into the main program.

Sources Palette



Sinks Palette

Block Screenshot



Stateflow

Delays in Stateflow using Temporal Logic

https://in.mathworks.com/help/stateflow/ug/using-temporal-logic-in-state-actions-and-transitions.html

Operator	Usage	Example
`after`	State action (on after)	`on after(5, CLK): status('on');`
`after`	Transition	`ROTATE[after(10, CLK)]`
`before`	State action (on before)	`on before(MAX, CLK): temp++;`
`before`	Transition	`ROTATE[before(10, CLK)]`
`at`	State action (on at)	`on at(10, CLK): status('on');`
`at`	Transition	`ROTATE[at(10, CLK)]`
`every`	State action (on every)	`on every(5, CLK): status('on');`
`temporalCount`	State action	`du: y = mm[temporalCount(tick)];`

StateFlow

What is a history junction?

history junction is used to store the last active sub state inside a state machine and it will overwrite the default active state.
so whenever the machine enters the state again,the last active state becomes the default state by which we assure that the last valid settings made are not lost

what are flow charts?

Flow charts or Flow Graphs are means of designing loop statements alike for,switch,case inside state machines.
we can design for loops,switch cases,if else loops,**customized** logic's using flow charts.when the system behavior involves more conditional logics involving if,switch logics it is better to design it using flow graphs

when flow charts are used and when state flow is used?

Flow Graphs are used in more loop based environment whereas stateflow involves more complex logic's.
But the Choice is based on the taste of the developer and the application

when to use stateflow and when to use simulink?

stateflow is used for discrete time systems and simulink is used for continuous time systems
But whatever written in C can be designed in both simulink and stateflow.
But stateflow increases model readability for discrete time system in comparison to designing the same in simulink

what are junctions?

when there are more than one condition involving a transition like

Time Partition Testing (TPT)

https://en.wikipedia.org/wiki/TPT_(software)

TPT – Time Partition Testing

Automated, reactive testing and verification for Simulink models and Simulink Real-Time

Highlights

- Model-based functional and closed-loop testing
- Automatic test harness creation, test execution, assessment, and reporting
- Testing models (MIL), code (SIL), and real-time applications (HIL)
- Testing according to safety norms (e.g., ISO 26262, IEC 61508)
- Traceability and impact analysis of requirements and tests
- Back-to-back analysis and coverage support

TPT (software)

From Wikipedia, the free encyclopedia
(Redirected from Time Partition Testing)

TPT (**time partition testing**) is a systematic test methodology for the automated software test and verification of embedded control systems or dataflow programs. TPT is specialised on testing and validation of embedded systems whose inputs and outputs can be represented as signals and is a dedicated method for testing continuous behaviour of systems.[1] Most control systems belong to this system class. The outstanding characteristic of control systems is the fact that they interact closely interlinked with a real world environment. Controllers need to observe their environment and react correspondingly to its behaviour.[2] The system works in an interactional cycle with its environment and is subject to temporal constraints. Testing these systems is to stimulate and to check the timing behaviour. Traditional functional testing methods use scripts – TPT uses model-based testing.

TPT combines a systematic and graphic modelling technique for test cases with a fully automated test execution in different environments and automatic test evaluation. TPT covers the following four test activities:

- test case modelling
- test execution in different environments (automated)
- test result analysis (test assessment (automated))
- test documentation (automated)
- test management

Vehicle Components

ADC

https://en.wikipedia.org/wiki/Analog-to-digital_converter

Explanation [edit]

An ADC converts a continuous-time and continuous-amplitude analog signal to a discrete-time and discrete-amplitude digital signal. The conversion involves quantization of the input, so it necessarily introduces a small amount of error or noise. Furthermore, instead of continuously performing the conversion, an ADC does the conversion periodically, sampling the input, limiting the allowable bandwidth of the input signal.

The performance of ADC is characterized by its bandwidth and its signal-to-noise ratio. The bandwidth of an ADC is characterized primarily by its sampling rate. The dynamic range of an ADC is influenced by many factors, including the resolution, linearity and accuracy (how well the quantization levels match the true analog signal), aliasing and jitter. The dynamic range of an ADC is often summarized in terms of its effective number of bits (ENOB), the number of bits of each measure it returns that are on average not noise. An ideal ADC has an ENOB equal to its resolution. ADCs are chosen to match the bandwidth and required signal-to-noise ratio of the signal to be quantized. If an ADC operates at a sampling rate greater than twice the bandwidth of the signal, then perfect reconstruction is possible given an ideal ADC and neglecting quantization error. The presence of quantization error limits the dynamic range of even an ideal ADC. However, if the dynamic range of the ADC exceeds that of the input signal, its effects may be neglected resulting in an essentially perfect digital representation of the input signal.

https://en.wikipedia.org/wiki/Successive_approximation_ADC

Sensors & Actuators

Sensor

https://en.wikipedia.org/wiki/Sensor

Output = Input/Sensitivity

Where Output => May Be Temperature
Input => May Be Voltage
Sensitivity => Constant for Monitoring the Same

Classification of measurement errors [edit]

A good sensor obeys the following rules:[citation needed]:

- it is sensitive to the measured property
- it is insensitive to any other property likely to be encountered in its application, and
- it does not influence the measured property.

An infrared sensor

Most sensors have a linear transfer function. The sensitivity is then defined as the ratio between the output signal and measured property. For example, if a sensor measures temperature and has a voltage output, the sensitivity is a constant with the units [V/K]. The sensitivity is the slope of the transfer function. Converting the sensor's electrical output (for example V) to the measured units (for example K) requires dividing the electrical output by the slope (or multiplying by its reciprocal). In addition, an offset is frequently added or subtracted. For example, -40 must be added to the output if 0 V output corresponds to -40 C input.

For an analog sensor signal to be processed, or used in digital equipment, it needs to be converted to a digital signal, using an analog-to-digital converter.

Actuator

https://en.wikipedia.org/wiki/Actuator

Actuator

From Wikipedia, the free encyclopedia

An **actuator** is a component of a machine that is responsible for moving and controlling a mechanism or system, for example by opening a valve. In simple terms, it is a "mover".

An actuator requires a control signal and a source of energy. The control signal is relatively low energy and may be electric voltage or current, pneumatic or hydraulic pressure, or even human power. Its main energy source may be an electric current, hydraulic fluid pressure, or pneumatic pressure. When it receives a control signal, an actuator responds by converting the signal's energy into mechanical motion.

An actuator is the mechanism by which a control system acts upon an environment. The control system can be simple (a fixed mechanical or electronic system), software-based (e.g. a printer driver, robot control system), a human, or any other input.[1]

Vehicle Dynamics

https://en.wikipedia.org/wiki/Vehicle_dynamics

Sensor Modeling

Sensor Modeling

Special Language for specifiying the Different Sensor Characteristics

https://en.wikipedia.org/wiki/SensorML

Steps for Sensor Modeling

Physical Method

1. Derive the Physical Equation from the Sensor Data Sheet
2. Model the Equation

Empirical Method

1. Derive the Data Oriented Mechanism Like Voltage vs Temperature
2. Model it Directly using Interpolation

Protocols

OnePager - All Protocols

Competing Technologies/Standards

Ethernet is the "new-comer" to the automotive market. The following technologies are actively being used in cars today:

- CAN (Controller Area Network) – Available since 1983, CAN is a shared serial bus running at up to 1Mbps. It was developed by Bosch and standardized in multiple ISO standards. It has the advantage of being fairly inexpensive and being very reliable. It has the disadvantages of relatively low bandwidth and being a shared media (rather than a switched network) where any transmission takes away from the total bandwidth. CAN is used in powertrain, chassis, and body electronics.
- LIN (Local Interconnect Network) – Available since 2001, LIN was developed by a consortium of automakers and technology partners. Like CAN, it is a serial bus. It runs at only 19,200 baud and requires only one shared wire (instead of the 2 for CAN). Unlike CAN, in which all nodes are "equal," LIN is a master-slave architecture. Its main advantage is that it is lower cost than CAN. LIN is often used for body electronics (mirrors, power seats, accessories) as it is inexpensive and the bandwidth requirements are very low.
- FlexRay – Available since 2005, FlexRay is a shared serial bus running at up to 10Mbps. It was developed by the FlexRay consortium, a group of semiconductor, automobile, and infrastructure providers. Unlike CAN, there is no build it error recovery – rather error handling is left to the application layer. It has the advantage of having higher bandwidth than CAN, but the disadvantage of higher cost and being a shared media. FlexRay is used in high-performance powertrain and safety (drive-by-wire, active suspension, adaptive cruise control).
- MOST (Media Oriented Systems Transport) – Available since around 2001, MOST has a ring architecture running at up to 50Mbps using either fiber or copper interconnects. Each ring can contain up to 64 MOST devices. MOST has the advantage or relatively high bandwidth (in the automotive market), but is very high cost. It is only used for camera or video connections.
- LVDS (Low Voltage Differential Signaling) – Defined in 1994, LVDS has been gaining use in the automotive market as a replacement for most. Unlike MOST/LIN/FlexRay/MOST, LVDS is a point-to-point bus (not a shared bus). It has the advantage of much lower cost than MOST and many auto-makers have started using this for camera and video data. The disadvantage is that each LVDS bus can only be used to interface to only one camera or video output.

Ethernet is the "new-comer" to the automotive market.

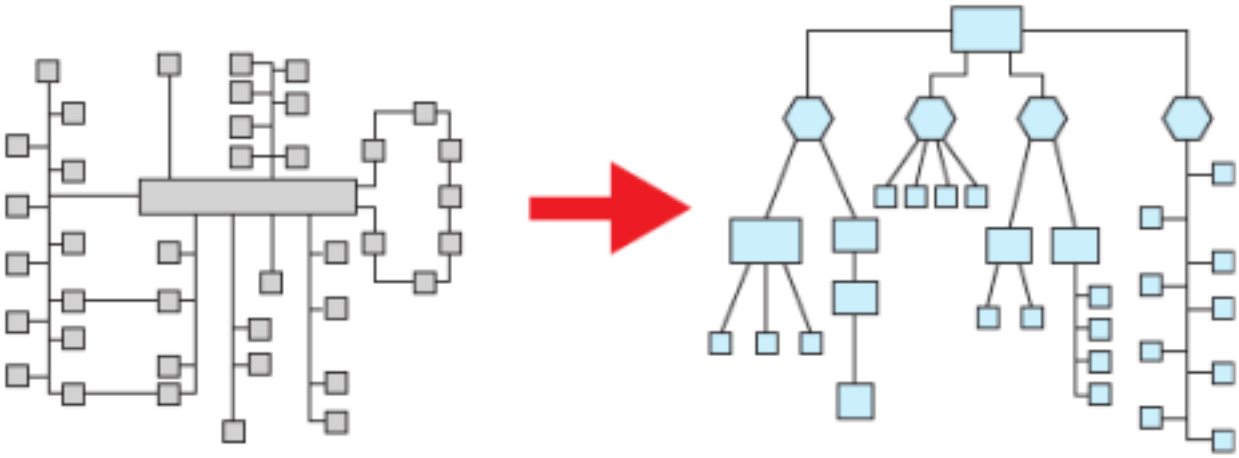
Automotive Ethernet

Domain	Description	End-to-End Latency Requirements	Bandwidth Requirements
Powertrain	Controls the components that generate power and transmit to the road.	< 10 us	Low
Chassis	Controls steering, brakes, suspension	< 10 us	Low
Body and Comfort	Radio, A/C, window, seat, and light controls	< 10 ms	Low
Driver Assistance and Driver Safety	Controls systems designed to increase safety	< 250 us or < 1 ms depending on system	20 – 100 Mbps per camera
Human-Machine Interface	Controls displays and other interfaces that interface with the driver or passengers	< 10 ms	Varies by system, but this growing

Why Ethernet was Not Used in Cars until Now

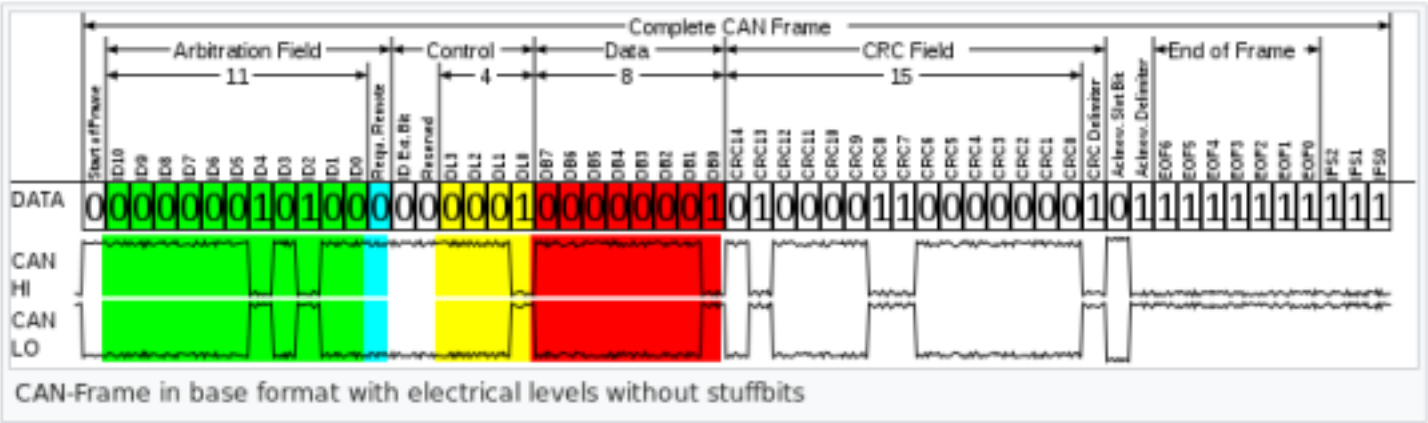
Even though Ethernet has existed for over 20 years, it could not be previously used in automobiles due to the following limitations:

1. Ethernet did not meet the OEM EMI/RFI requirements for the automotive market. 100Mbps (and above) Ethernet have too much RF "noise," and Ethernet is also susceptible to "alien" noise from other devices in a car.
2. Ethernet could not guarantee latency down to the low microsecond range. This was required to replace communication to any sensor/control that needed fast reaction time.
3. Ethernet did not have a way to control bandwidth allocation to different streams so it could not be used to transmit shared data from multiple types of sources.
4. Ethernet did not have a way of synchronizing time between devices and having multiple devices sample data at the same time.



CAN

https://en.wikipedia.org/wiki/CAN_bus



CAN-Frame in base format with electrical levels without stuffbits

Standard Frame Format

Field name	Length (bits)	Purpose
Start-of-frame	1	Denotes the start of frame transmission
Identifier (green)	11	A (unique) identifier which also represents the message priority
Remote transmission request (RTR) (blue)	1	Must be dominant (0) for data frames and recessive (1) for remote request frames (see Remote Frame, below)
Identifier extension bit (IDE)	1	Must be dominant (0) for base frame format with 11-bit identifiers
Reserved bit (r0)	1	Reserved bit. Must be dominant (0), but accepted as either dominant or recessive.
Data length code (DLC) (yellow)	4	Number of bytes of data (0–8 bytes)[a]
Data field (red)	0–64 (0-8 bytes)	Data to be transmitted (length in bytes dictated by DLC field)
CRC	15	Cyclic redundancy check
CRC delimiter	1	Must be recessive (1)
ACK slot	1	Transmitter sends recessive (1) and any receiver can assert a dominant (0)
ACK delimiter	1	Must be recessive (1)
End-of-frame (EOF)	7	Must be recessive (1)

Extended Frame Format

Field name	Length (bits)	Purpose
Start-of-frame	1	Denotes the start of frame transmission
Identifier A (green)	11	First part of the (unique) identifier which also represents the message priority
Substitute remote request (SRR)	1	Must be recessive (1)
Identifier extension bit (IDE)	1	Must be recessive (1) for extended frame format with 29-bit identifiers
Identifier B (green)	18	Second part of the (unique) identifier which also represents the message priority
Remote transmission request (RTR) (blue)	1	Must be dominant (0) for data frames and recessive (1) for remote request frames (see Remote Frame, below)
Reserved bits (r1, r0)	2	Reserved bits which must be set dominant (0), but accepted as either dominant or recessive
Data length code (DLC) (yellow)	4	Number of bytes of data (0–8 bytes)[a]
Data field (red)	0–64 (0-8 bytes)	Data to be transmitted (length dictated by DLC field)
CRC	15	Cyclic redundancy check
CRC delimiter	1	Must be recessive (1)
ACK slot	1	Transmitter sends recessive (1) and any receiver can assert a dominant (0)
ACK delimiter	1	Must be recessive (1)
End-of-frame (EOF)	7	Must be recessive (1)

SAE J1939

https://en.wikipedia.org/wiki/SAE_J1939

Definition [edit]

SAE J1939 defines five layers in the seven-layer OSI network model, and this includes the Controller Area Network (CAN) ISO 11898 specification (using only the 29-bit/"extended" identifier) for the physical and data-link layers. Under J1939/11 and J1939/15, the data rate is specified as 250 kbit/s, with J1939/14 specifying 500 kbit/s. The session and presentation layers are not part of the specification. The later use of CAN FD is currently discussed.[when?]

All J1939 packets, except for the request packet, contain eight bytes of data and a standard header which contains an index called Parameter Group Number (PGN), which is embedded in the message's 29-bit identifier. A PGN identifies a message's function and associated data. J1939 attempts to define standard PGNs to encompass a wide range of automotive, agricultural, marine and off-road vehicle purposes. A range of PGNs ($00FF00_{16}$ through $00FFFF_{16}$, inclusive) is reserved for proprietary use. PGNs define the data which is made up of a variable number of Suspect Parameter Number (SPN) elements defined for unique data. For example, there exists a predefined SPN for engine RPM.

Programming

Miscellaneous

Continuous Integration

Travis - Based in Berlin, Germany

https://en.wikipedia.org/wiki/Travis_CI

1. Write an Yaml File with your Changes.
2. Once the GitHub Repo is Pushed, GitHub detects the Changes and runs your automated tests based on the configuration from the Yaml File
3. Throws Results to Each Developer Branch

Travis CI is a hosted, distributed[2] continuous integration service used to build and test software projects hosted at GitHub.[3]

Open source projects may be tested at no charge via travis-ci.org. Private projects may be tested at travis-ci.com on a fee basis. TravisPro provides custom deployments of a proprietary version on the customer's own hardware.

Although the source is technically free software and available piecemeal on GitHub under permissive licenses, the company notes that it is unlikely that casual users could successfully integrate it on their own platforms.[4]

Jenkins - Written in Java. Self Hosted Automation Server with Multiple Features

1. Works as Both CI/CD
2. Multiple Features Exists like sending mail on Build Fails, Test Fails etc.,

https://jenkins.io/

Jenkins is an open source automation server written in Java. Jenkins helps to automate the non-human part of the software development process, with continuous integration and facilitating technical aspects of continuous delivery. It is a server-based system that runs in servlet containers such as Apache Tomcat. It supports version control tools, including AccuRev, CVS, Subversion, Git, Mercurial, Perforce, ClearCase and RTC, and can execute Apache Ant, Apache Maven and sbt based projects as well as arbitrary shell scripts and Windows batch commands. The creator of Jenkins is Kohsuke Kawaguchi.[3] Released under the MIT License, Jenkins is free software.[4]

Builds can be triggered by various means, for example by commit in a version control system, by scheduling via a cron-like mechanism and by requesting a specific build URL. It can also be triggered after the other builds in the queue have completed. Jenkins functionality can be extended with plugins.

The Jenkins project was renamed after a dispute with Oracle, and its fork, Hudson, continued to be developed by Oracle for a time before being donated to the Eclipse Foundation.

On Linux, BSD, and Mac OS (Unix-like) systems, the `sh` step is used to execute a shell command in a Pipeline.

Jenkinsfile (Declarative Pipeline)

```
pipeline {
    agent any
    stages {
        stage('Build') {
            steps {
                sh 'echo "Hello World"'
                sh '''
                    echo "Multiline shell steps works too"
                    ls -lah
                '''
            }
        }
    }
}
```

Some Other CIs

1. Hudson

OOP Terminologies

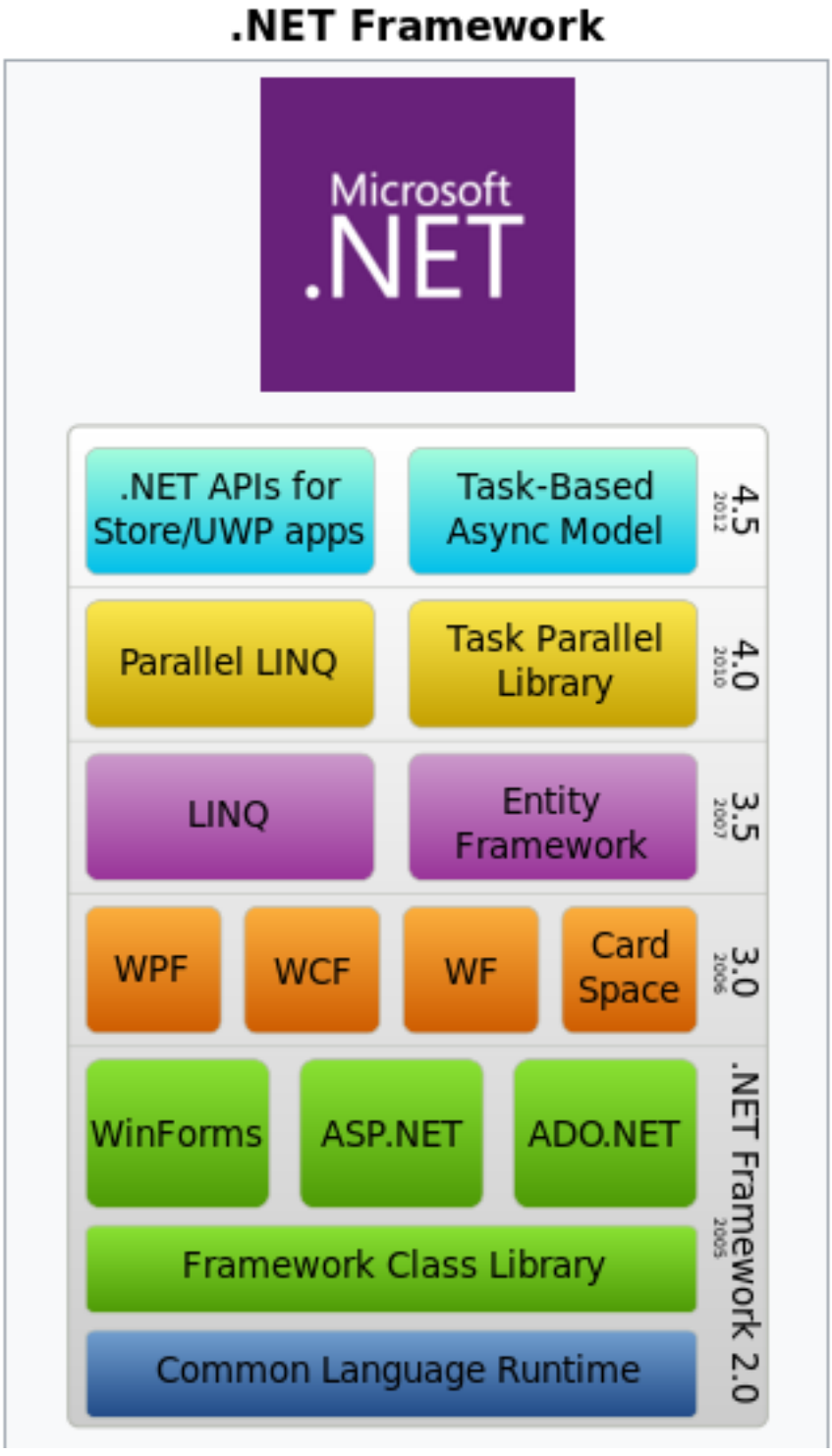
https://en.wikipedia.org/wiki/Object-oriented_programming

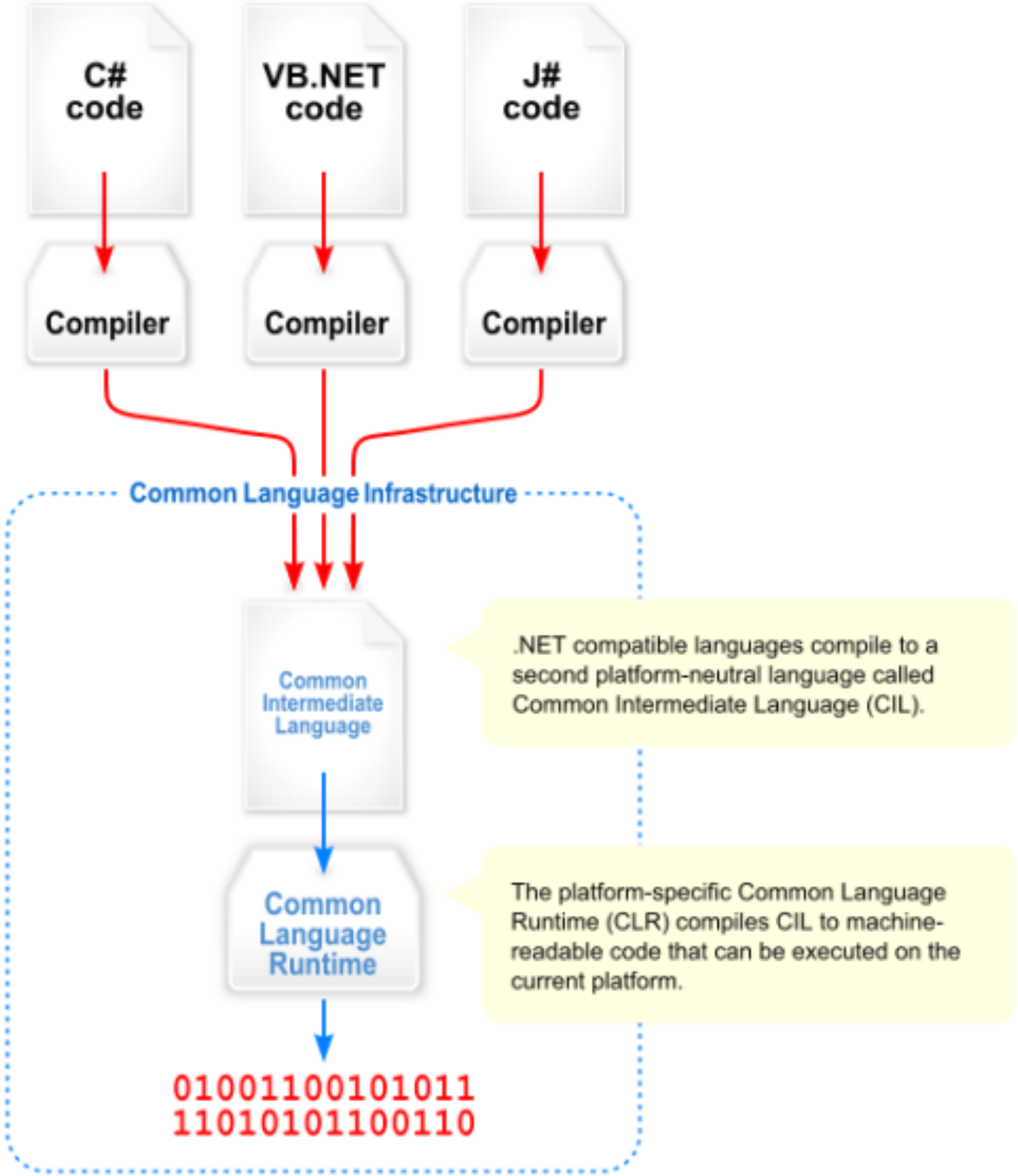
Programming

Functions called from the Object Instances are called Methods

.NET Framework

https://en.wikipedia.org/wiki/.NET_Framework




C# code
VB.NET code
J# code
Compiler
Compiler
Compiler
Common Language Infrastructure
Common Intermediate Language
.NET compatible languages compile to a second platform-neutral language called Common Intermediate Language (CIL).
Common Language Runtime
The platform-specific Common Language Runtime (CLR) compiles CIL to machine-readable code that can be executed on the current platform.
01001100101011
11010101100110

".NET" redirects here. For the Internet domain, see .net. For other uses, see .NET (disambiguation).

.NET Framework (pronounced *dot net*) is a software framework developed by Microsoft that runs primarily on Microsoft Windows. It includes a large class library named Framework Class Library (FCL) and provides language interoperability (each language can use code written in other languages) across several programming languages. Programs written for .NET Framework execute in a software environment (in contrast to a hardware environment) named Common Language Runtime (CLR), an application virtual machine that provides services such as security, memory management, and exception handling. (As such, computer code written using .NET Framework is called "managed code".) FCL and CLR together constitute .NET Framework.

FCL provides user interface, data access, database connectivity, cryptography, web application development, numeric algorithms, and network communications. Programmers produce software by combining their source code with .NET Framework and other libraries. The framework is intended to be used by most new applications created for the Windows platform. Microsoft also produces an integrated development environment largely for .NET software called Visual Studio.

Design principle [edit]

Interoperability [edit]

Because computer systems commonly require interaction between newer and older applications, .NET Framework provides means to access functions implemented in newer and older programs that execute outside .NET environment. Access to Component Object Model (COM) components is provided in `System.Runtime.InteropServices` and `System.EnterpriseServices` namespaces of the framework. Access to other functions is via Platform Invocation Services (P/Invoke). Access to .NET functions from native applications is via reverse P/Invoke function.

Language independence [edit]

.NET Framework introduces a Common Type System (CTS) that defines all possible data types and programming constructs supported by CLR and how they may or may not interact with each other conforming to CLI specification. Because of this feature, .NET Framework supports the exchange of types and object instances between libraries and applications written using any conforming .NET language.

Type safety [edit]

CTS and the CLR used in .NET Framework also enforce type safety. This prevents ill-defined casts, wrong method invocations, and memory size issues when accessing an object. This also makes most CLI languages statically typed (with or without type inference). However, starting with .NET Framework 4.0, the Dynamic Language Runtime extended the CLR, allowing dynamically typed languages to be implemented atop the CLI.

Portability [edit]

While Microsoft has never implemented the full framework on any system except Microsoft Windows, it has engineered the framework to be cross-platform,[48] and implementations are available for other operating systems (see Silverlight and § Alternative implementations). Microsoft submitted the specifications for CLI (which includes the core class libraries, CTS, and CIL),[49][50][51] C#,[52] and C++/CLI[53] to both Ecma International (ECMA) and International Organization for Standardization (ISO), making them available as official standards. This makes it possible for third parties to create compatible implementations of the framework and its languages on other platforms.

Security [edit]

Alternative implementations [edit]

.NET Framework is the predominant implementation of .NET technologies. Other implementations for parts of the framework exist. Although the runtime engine is described by an ECMA-ISO specification, other implementations of it may be encumbered by patent issues; ISO standards may include the disclaimer, "Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights."[67] It is harder to develop alternatives to FCL, which is not described by an open standard and may be subject to copyright restrictions. Also, parts of FCL have Windows-specific functions and behavior, so implementation on non-Windows platforms can be problematic.

Some alternative implementations of parts of the framework are listed here.

- .NET Micro Framework is a .NET platform for extremely resource-constrained devices. It includes a small version of CLR and supports development in C# (though some developers were able to use VB.NET,[68] albeit with an amount of hacking, and with limited functionalities) and debugging (in an emulator or on hardware), both using Microsoft Visual Studio. It also features a subset of .NET Framework Class Library (about 70 classes with about 420 methods), a GUI framework loosely based on WPF, and additional libraries specific to embedded applications.
- .NET Core is an alternative Microsoft implementation of the managed code framework; it has similarities with .NET Framework and even shares some API, but is designed based on different sets of principles: it is cross-platform and free and open-source.
- Mono is an implementation of CLI and FCL, and provides added functions. It is dual-licensed as free and proprietary software. It includes support for ASP.NET, ADO.NET, and Windows Forms libraries for a wide range of architectures and operating systems. It also includes C# and VB.NET compilers.
- Portable.NET (part of DotGNU) provides an implementation of CLI, parts of FCL, and a C# compiler. It supports a variety of CPUs and operating systems. The project was discontinued, with the last stable release in 2009.
- Microsoft Shared Source Common Language Infrastructure is a non-free implementation of CLR. However, the last version runs on Windows XP SP2 only, and has not been updated since 2006. Thus, it does not contain all features of version 2.0 of .NET Framework.
- CrossNet[69] is an implementation of CLI and parts of FCL. It is free software using an open source MIT License.

Clojure

https://en.wikipedia.org/wiki/Clojure

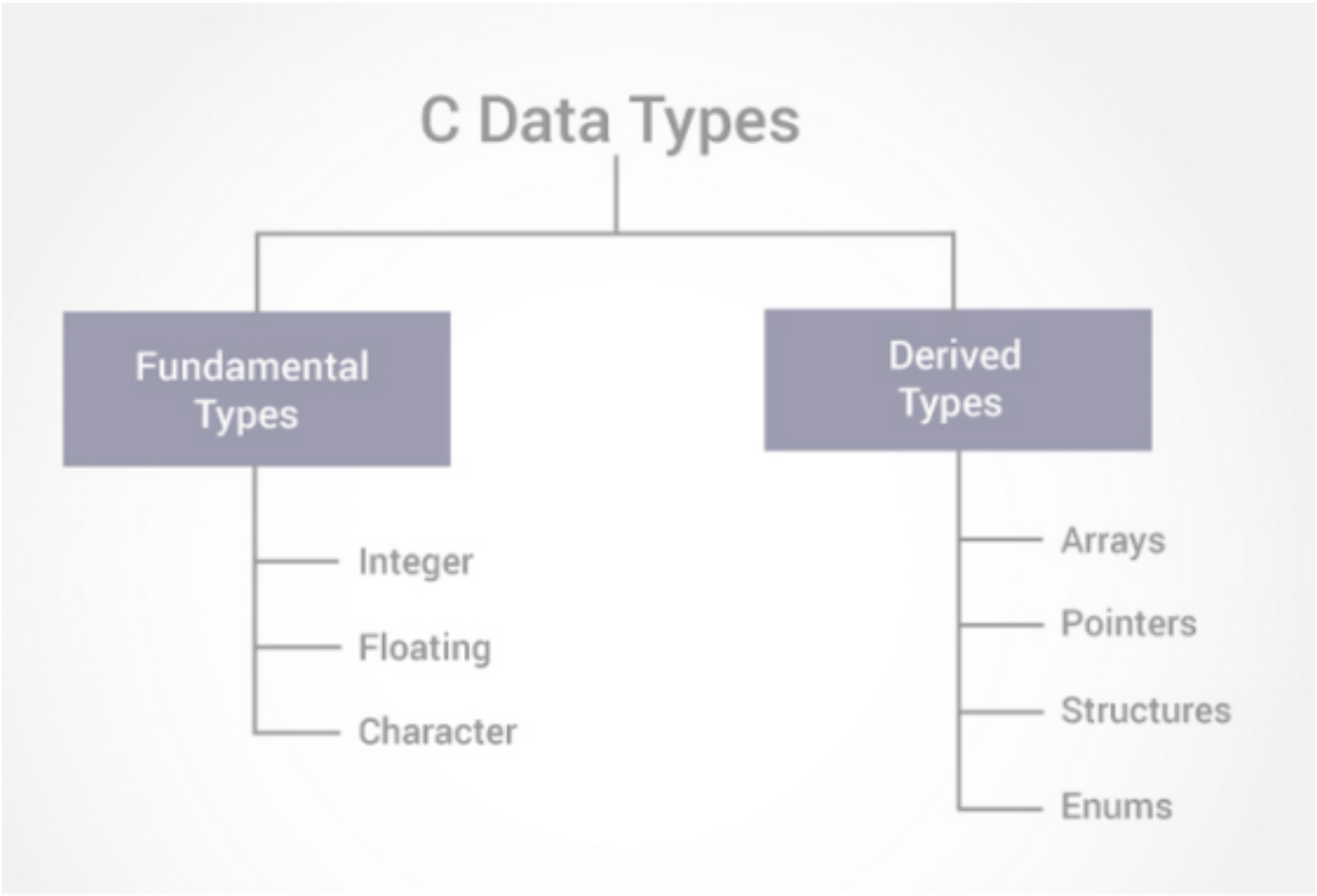
Clojure (/ˈkloʊʒʊr/, like "closure"[6]) is a dialect of the Lisp programming language.[7] Clojure is a general-purpose programming language with an emphasis on functional programming.[8] It runs on the Java virtual machine and the Common Language Runtime.[9] There is a dialect, developed in lockstep with Clojure, called ClojureScript,[10] that compiles to ECMAScript 3.[11] Like other Lisps, Clojure treats code as data and has a macro system.[12] The current development process is community-driven,[13] overseen by Rich Hickey as its benevolent dictator for life (BDFL).[14]

Clojure encourages immutability and immutable data structures. While its type system is entirely dynamic, recent efforts have also sought the implementation of gradual typing.[15] Clojure encourages programmers to be explicit about managing state and identity.[16] This focus on programming with immutable values and explicit progression-of-time constructs is intended to facilitate developing more robust programs, especially multithreaded ones.[17][18]

Clojure is used in industry by firms such as Funding Circle,[19] Walmart,[20] Puppet,[21] and other large software firms.[22] Commercial support for Clojure is provided by Cognitect.[22] Annual Clojure conferences are organised every year across the globe, the most famous of them being Clojure/conj (US east coast),[23] Clojure/West (US west coast),[24] and EuroClojure (Europe).[25]

C

Datatypes in C



Type	Storage size	Value range
char	1 byte	-128 to 127 or 0 to 255
unsigned char	1 byte	0 to 255
signed char	1 byte	-128 to 127
int	2 or 4 bytes	-32,768 to 32,767 or -2,147,483,648 to 2,147,483,647
unsigned int	2 or 4 bytes	0 to 65,535 or 0 to 4,294,967,295
short	2 bytes	-32,768 to 32,767
unsigned short	2 bytes	0 to 65,535
long	4 bytes	-2,147,483,648 to 2,147,483,647
unsigned long	4 bytes	0 to 4,294,967,295

Type	Storage size	Value range	Precision
float	4 byte	1.2E-38 to 3.4E+38	6 decimal places
double	8 byte	2.3E-308 to 1.7E+308	15 decimal places
long double	10 byte	3.4E-4932 to 1.1E+4932	19 decimal places

Type	Explanation	Format Specifier
char	Smallest addressable unit of the machine that can contain basic character set. It is an integer type. Actual type can be either signed or unsigned. It contains `CHAR_BIT` bits.[3]	%c
signed char	Of the same size as `char`, but guaranteed to be signed. Capable of containing **at least** the [−127, +127] range;[3][4]	%c (or %hhi for numerical output)
unsigned char	Of the same size as `char`, but guaranteed to be unsigned. Contains **at least** the [0, 255] range.[5]	%c (or %hhu for numerical output)
short short int signed short signed short int	*Short* signed integer type. Capable of containing **at least** the [−32,767, +32,767] range;[3][4] thus, it is at least 16 bits in size. The negative value is −32767 (not −32768) due to the one's-complement and sign-magnitude representations allowed by the standard, though the two's-complement representation is much more common.[6]	%hi
unsigned short unsigned short int	*Short* unsigned integer type. Contains **at least** the [0, 65,535] range;[3][4]	%hu
int signed signed int	Basic signed integer type. Capable of containing **at least** the [−32,767, +32,767] range;[3][4] thus, it is at least 16 bits in size.	%i or %d
unsigned unsigned int	Basic unsigned integer type. Contains **at least** the [0, 65,535] range;[3][4]	%u
long long int signed long signed long int	*Long* signed integer type. Capable of containing **at least** the [−2,147,483,647, +2,147,483,647] range;[3][4] thus, it is at least 32 bits in size.	%li
unsigned long unsigned long int	*Long* unsigned integer type. Capable of containing **at least** the [0, 4,294,967,295] range;[3][4]	%lu
long long long long int signed long long signed long long int	*Long long* signed integer type. Capable of containing **at least** the [−9,223,372,036,854,775,807, +9,223,372,036,854,775,807] range;[3][4] thus, it is at least 64 bits in size. Specified since the C99 version of the standard.	%lli
unsigned long long unsigned long long int	*Long long* unsigned integer type. Contains **at least** the [0, +18,446,744,073,709,551,615] range;[3][4] Specified since the C99 version of the standard.	%llu
float	Real floating-point type, usually referred to as a single-precision floating-point type. Actual properties unspecified (except minimum limits), however on most systems this is the IEEE 754 single-precision binary floating-point format (32 bits). This format is required by the optional Annex F "IEC 60559 floating-point arithmetic".	for formatted input: %f %F for digital notation, or %g %G, or %e %E %a %A for scientific notation[7]
double	Real floating-point type, usually referred to as a double-precision floating-point type. Actual properties unspecified (except minimum limits), however on most systems this is the IEEE 754 double-precision binary floating-point format (64 bits). This format is required by the optional Annex F "IEC 60559 floating-point arithmetic".	%lf %lF %lg %lG %le %lE %la %lA;[7] for formatted output, the length modifier l is optional.
long double	Real floating-point type, usually mapped to an extended precision floating-point number format. Actual properties unspecified. It can be either x86 extended-precision floating-point format (80 bits, but typically 96 bits or 128 bits in memory with padding bytes), the non-IEEE "double-double" (128 bits), IEEE 754 quadruple-precision floating-point format (128 bits), or the same as double. See the article on long double	%Lf %LF %Lg %LG %Le %LE

C vs Embedded C

https://www.quora.com/What-is-the-difference-between-C-and-Embedded-C

- C is generally used for desktop computers, while embedded C is for microcontroller based applications.
- C can use the resources of a desktop PC like memory, OS, etc. While, embedded C has to use with the limited resources, such as RAM, ROM, I/Os on an embedded processor.
- Embedded C includes extra features over C, such as fixed point types, multiple memory areas, and I/O register mapping.
- Compilers for C (ANSI C) typically generate OS dependant executables. Embedded C requires compilers to create files to be downloaded to the microcontrollers/microprocessors where it needs to run.

> Embedded systems are programmed using different type of languages:
>
> - Machine Code
> - Low level language, i.e., assembly
> - High level language like C, C++, Java, Ada, etc.
> - Application level language like Visual Basic, scripts, Access, etc.

Use of **C in embedded systems** is driven by following advantages

- It is small and reasonably simpler to learn, understand, program and debug.
- C Compilers are available for almost all embedded devices in use today, and there is a large pool of experienced C programmers.
- Unlike assembly, C has advantage of processor-independence and is not specific to any particular microprocessor/ microcontroller or any system. This makes it convenient for a user to develop programs that can run on most of the systems.
- As C combines functionality of assembly language and features of high level languages, C is treated as a 'middle-level computer language' or 'high level assembly language'
- It is fairly efficient
- It supports access to I/O and provides ease of management of large embedded projects.

Many of these advantages are offered by other languages also, but what sets C apart from others like Pascal, FORTRAN, etc. is the fact that it is a middle level language; it provides direct hardware control without sacrificing benefits of high level languages.

- A set of language extension for C is called Embedded C whereas desktop computer language is generally called C programming language.
- C directly run program from OS terminal whereas embedded C needs to create the file first then download to the embedded system where the compiling process is carried out.
- OS system is must for C programming whereas it's an option for Embedded C.
- See output on your desktop with C programming whereas no output can be observed on desktop with Embedded C, i.e. Embedded C runs in real time constraints.
- Programming languages like C++, JavaScript, Perl, Python, and many more are directly or indirectly influenced by C language whereas Embedded C is developed only for the required microprocessor/microcontroller.
- Embedded C is used for microcontrollers like TV, washing machines, etc. whereas C finds applications in simple yet logical programs, OS based software, etc.
- Based on microcontroller or processor, Embedded C comes with different formats while C programming comes with free-format source code.
- As mentioned before, Embedded C has limited source constraints like limited RAM/ROM etc. whereas C can make use of all computer resources.
- No data can be input in embedded C while running, due to its predefined data whereas C can easily intake program data while programming.

C Programming Questions

Keywords

Keywords in C Language

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
continue	for	signed	void
do	if	static	while
default	goto	sizeof	volatile
const	float	short	unsigned

COM

https://en.wikipedia.org/wiki/Component_Object_Model

Component Object Model (**COM**) is a binary-interface standard for software components introduced by Microsoft in 1993. It is used to enable inter-process communication object creation in a large range of programming languages. COM is the basis for several other Microsoft technologies and frameworks, including OLE, OLE Automation, Browser Helper Object, ActiveX, COM+, DCOM, the Windows shell, DirectX, UMDF and Windows Runtime. The essence of COM is a language-neutral way of implementing objects that can be used in environments different from the one in which they were created, even across machine boundaries. For well-authored components, COM allows reuse of objects with no knowledge of their internal implementation, as it forces component implementers to provide well-defined interfaces that are separated from the implementation. The different allocation semantics of languages are accommodated by making objects responsible for their own creation and destruction through reference-counting. Type conversion casting between different interfaces of an object is achieved through the `QueryInterface` method. The preferred method of "inheritance" within COM is the creation of sub-objects to which method "calls" are delegated.

COM is an interface technology defined and implemented as standard only on Microsoft Windows and Apple's Core Foundation 1.3 and later plug-in application programming interface (API).[1] The latter only implements a subset of the whole COM interface.[2] For some applications, COM has been replaced at least to some extent by the Microsoft .NET framework, and support for Web Services through the Windows Communication Foundation (WCF). However, COM objects can be used with all .NET languages through .NET COM Interop. Networked DCOM uses binary proprietary formats, while WCF encourages the use of XML-based SOAP messaging. COM is very similar to other component software interface technologies, such as CORBA and Enterprise JavaBeans, although each has its own strengths and weaknesses. Unlike C++, COM provides a stable application binary interface (ABI) that does not change between compiler releases.[3] This makes COM interfaces attractive for object-oriented C++ libraries that are to be used by clients compiled using different compiler versions.

Functional Programming

https://en.wikipedia.org/wiki/Functional_programming

Functional programming

From Wikipedia, the free encyclopedia

For subroutine-oriented programming, see Procedural programming.

In computer science, **functional programming** is a programming paradigm—a style of building the structure and elements of computer programs—that treats computation as the evaluation of mathematical functions and avoids changing-state and mutable data. It is a declarative programming paradigm, which means programming is done with expressions[1] or declarations[2] instead of statements. In functional code, the output value of a function depends only on the arguments that are passed to the function, so calling a function *f* twice with the same value for an argument *x* produces the same result *f(x)* each time; this is in contrast to procedures depending on a local or global state, which may produce different results at different times when called with the same arguments but a different program state. Eliminating side effects, i.e., changes in state that do not depend on the function inputs, can make it much easier to understand and predict the behavior of a program, which is one of the key motivations for the development of functional programming.

Higher Order Functions

Python [edit]

Further information: Python (programming language)

```
>>> def twice(f):
...     return lambda x: f(f(x))

>>> def plusthree(x):
...     return x + 3

>>> g = twice(plusthree)

>>> g(7)
13
```

Type Inference

The signature of this function definition, `int addone(int x)`, declares that `addone` is a function that takes one argument, an integer, and returns an integer. `int result;` declares that the local variable `result` is an integer. In a hypothetical language supporting type inference, the code might be written like this instead:

```
addone(x) {
    var result;  /* inferred-type variable result */
    var result2; /* inferred-type variable result #2 */

    result = x + 1;
    result2 = x + 1.0;  /* this line won't work (in the proposed language) */
    return result;
}
```

This is identical to how code is written in the language Dart, except that it is subject to some added constraints as described below. It would be possible to *infer* the types of all the variables at compile time. In the example above, the compiler would infer that `result` and `x` have type integer since the constant `1` is type integer, and hence that `addone` is a function `int -> int`. The variable `result2` isn't used in a legal manner, so it wouldn't have a type.

Pure Functions

Pure functions [edit]

Pure functions (or expressions) have no side effects (memory or I/O). This means that pure functions have several useful properties, many of which can be used to optimize the code:

- If the result of a pure expression is not used, it can be removed without affecting other expressions.
- If a pure function is called with arguments that cause no side-effects, the result is constant with respect to that argument list (sometimes called referential transparency), i.e., if calling the pure function again with the same arguments returns the same result. (This can enable caching optimizations such as memoization.)
- If there is no data dependency between two pure expressions, their order can be reversed, or they can be performed in parallel and they cannot interfere with one another (in other terms, the evaluation of any pure expression is thread-safe).
- If the entire language does not allow side-effects, then any evaluation strategy can be used; this gives the compiler freedom to reorder or combine the evaluation of expressions in a program (for example, using deforestation).

Recursion

Recursion [edit]

Main article: Recursion (computer science)

Iteration (looping) in functional languages is usually accomplished via recursion. Recursive functions invoke themselves, letting an operation be repeated until it reaches the base case. Though some recursion requires maintaining a stack, tail recursion can be recognized and optimized by a compiler into the same code used to implement iteration in imperative languages. The Scheme language standard requires implementations to recognize and optimize tail recursion. Tail recursion optimization can be implemented by transforming the program into continuation passing style during compiling, among other approaches.

Strict vs Non Strict Evaluation

Strict versus non-strict evaluation [edit]

Main article: Evaluation strategy

Functional languages can be categorized by whether they use *strict (eager)* or *non-strict (lazy)* evaluation, concepts that refer to how function arguments are processed when an expression is being evaluated. The technical difference is in the denotational semantics of expressions containing failing or divergent computations. Under strict evaluation, the evaluation of any term containing a failing subterm fails. For example, the expression:

```
print length([2+1, 3*2, 1/0, 5-4])
```

fails under strict evaluation because of the division by zero in the third element of the list. Under lazy evaluation, the length function returns the value 4 (i.e., the number of items in the list), since evaluating it does not attempt to evaluate the terms making up the list. In brief, strict evaluation always fully evaluates function arguments before invoking the function. Lazy evaluation does not evaluate function arguments unless their values are required to evaluate the function call itself.

Javascript

Perl

https://www.cheatography.com/mishin/cheat-sheets/perl-reference-card/

Python

https://en.wikipedia.org/wiki/Python_(programming_language)

Python (programming language)

From Wikipedia, the free encyclopedia

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales.[26]

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.[27]

Python interpreters are available for many operating systems. CPython, the reference implementation of Python, is open source software[28] and has a community-based development model, as do nearly all of its variant implementations. CPython is managed by the non-profit Python Software Foundation.

DataTypes

Type	mutable	Description	Syntax example
`bool`	immutable	Boolean value	`True` `False`

`bytearray`	mutable	Sequence of bytes	`bytearray(b'Some ASCII')`
			`bytearray(b"Some ASCII")`
			`bytearray([119, 105, 107, 105])`

`bytes`	immutable	Sequence of bytes	`b'Some ASCII'`
			`b"Some ASCII"`
			`bytes([119, 105, 107, 105])`

`complex`	immutable	Complex number with real and imaginary parts	`3+2.7j`
`dict`	mutable	Associative array (or dictionary) of key and value pairs; can contain mixed types (keys and values), keys must be a hashable type	`{'key1': 1.0, 3: False}`
`ellipsis`		An ellipsis placeholder to be used as an index in NumPy arrays	`...`
`float`	immutable	Floating point number, system-defined precision	`3.1415927`
`frozenset`	immutable	Unordered set, contains no duplicates; can contain mixed types, if hashable	`frozenset([4.0, 'string', True])`
`int`	immutable	Integer of unlimited magnitude[76]	`42`
`list`	mutable	List, can contain mixed types	`[4.0, 'string', True]`
`set`	mutable	Unordered set, contains no duplicates; can contain mixed types, if hashable	`{4.0, 'string', True}`

`str`	immutable	A character string: sequence of Unicode codepoints	`'Wikipedia'`
			`"Wikipedia"`
			`"""Spanning multiple`

Defining Class and Modules in Python

https://www.digitalocean.com/community/tutorials/how-to-write-modules-in-python-3

hello.py

```
# Define a function
def world():
    print("Hello, World!")
```

main_program.py

```
# Import hello module
import hello


# Call function
hello.world()
```

Any Folder with __init__.py is considered as a module.
So Append this path to the overall python sys.append Path.
And then natively you should be able to access the modules/functions in the Python REPL

Then you can import `hello`:

```
>>> import hello
>>> hello.helloworld()
'hello'
>>>
```

To group many `.py` files put them in a folder. Any folder with an `__init__.py` is considered a module by python and you can call them a package

```
|-HelloModule
  |_ __init__.py
  |_ hellomodule.py
```

You can go about with the import statement on your module the usual way.

For more information, see 6.4. Packages.

R

https://en.wikipedia.org/wiki/R_(programming_language)

R (programming language)

From Wikipedia, the free encyclopedia

R is a programming language and free software environment for statistical computing and graphics that is supported by the R Foundation for Statistical Computing.[6] The R language is widely used among statisticians and data miners for developing statistical software[7] and data analysis.[8] Polls, surveys of data miners, and studies of scholarly literature databases show that R's popularity has increased substantially in recent years.[9] As of May 2018, R ranks 11th in the TIOBE index, a measure of popularity of programming languages.[10]

R is a GNU package.[11] The source code for the R software environment is written primarily in C, Fortran, and R.[12] R is freely available under the GNU General Public License, and pre-compiled binary versions are provided for various operating systems. While R has a command line interface, there are several graphical front-ends [13] and integrated development environments[14] available.

Basic syntax [edit]

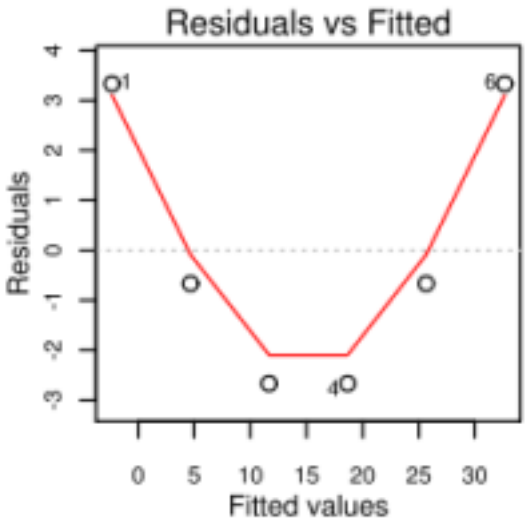
The following examples illustrate the basic syntax of the language and use of the command-line interface.

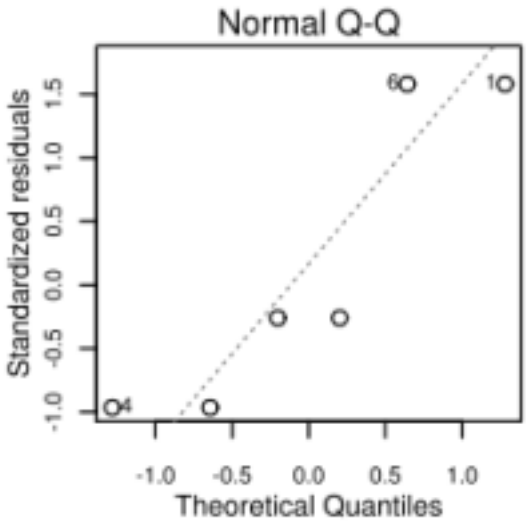
In R, the generally preferred[100] assignment operator is an arrow made from two characters `<-`, although `=` can usually be used instead.[101]

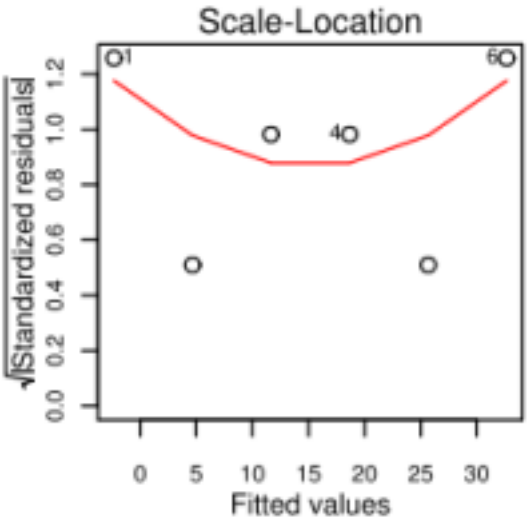
```
> x <- c(1, 2, 3, 4, 5, 6)   # Create ordered collection (vector)
> y <- x^2                   # Square the elements of x
> print(y)                   # print (vector) y
[1]  1  4  9 16 25 36
> mean(y)                    # Calculate average (arithmetic mean) of (vector) y; result is scalar
[1] 15.16667
> var(y)                     # Calculate sample variance
[1] 178.9667
> lm_1 <- lm(y ~ x)          # Fit a linear regression model "y = B0 + (B1 * x)"
                             # store the results as lm_1
> print(lm_1)                # Print the model from the (linear model object) lm_1

Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)            x
     -9.333        7.000
```

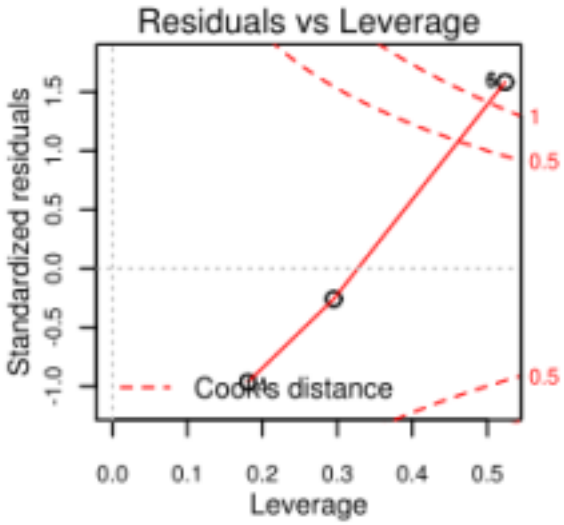
```
> summary(lm_1)              # Compute and print statistics for the fit
                             # of the (linear model object) lm_1

Call:
lm(formula = y ~ x)

Residuals:
1       2       3       4       5       6
3.3333 -0.6667 -2.6667 -2.6667 -0.6667  3.3333

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -9.3333     2.8441  -3.282 0.030453 *
x             7.0000     0.7303   9.585 0.000662 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.055 on 4 degrees of freedom
Multiple R-squared: 0.9583, Adjusted R-squared: 0.9478
F-statistic: 91.88 on 1 and 4 DF,  p-value: 0.000662

> par(mfrow = c(2, 2))       # Request 2x2 plot layout
```

```
> par(mfrow = c(2, 2))      # Request 2x2 plot layout
> plot(lm_1)                # Diagnostic plot of regression model
```









Scala

https://en.wikipedia.org/wiki/Scala_(programming_language)

Scala (programming language)

From Wikipedia, the free encyclopedia

Scala (/ˈskɑːlɑː/ *SKAH-lah*)[9] is a general-purpose programming language providing support for functional programming and a strong static type system. Designed to be concise,[10] many of Scala's design decisions aimed to address criticisms of Java.[8]

Scala source code is intended to be compiled to Java bytecode, so that the resulting executable code runs on a Java virtual machine. Scala provides language interoperability with Java, so that libraries written in both languages may be referenced directly in Scala or Java code.[11] Like Java, Scala is object-oriented, and uses a curly-brace syntax reminiscent of the C programming language. Unlike Java, Scala has many features of functional programming languages like Scheme, Standard ML and Haskell, including currying, type inference, immutability, lazy evaluation, and pattern matching. It also has an advanced type system supporting algebraic data types, covariance and contravariance, higher-order types (but not higher-rank types), and anonymous types. Other features of Scala not present in Java include operator overloading, optional parameters, named parameters, and raw strings. Conversely, a feature of Java not in Scala is checked exceptions, which have proved controversial.

The name Scala is a portmanteau of *scalable* and *language*, signifying that it is designed to grow with the demands of its users.[12]

Examples are:

- No distinction between statements and expressions
- Type inference
- Anonymous functions with capturing semantics (i.e., closures)
- Immutable variables and objects
- Lazy evaluation
- Delimited continuations (since 2.8)
- Higher-order functions
- Nested functions
- Currying
- Pattern matching
- Algebraic data types (through *case classes*)
- Tuples

VBA vs VB vs VBScript

VBA - No “exe” can be Produced where as VB with Visual Studio Support and .NET shall produce “exe”

http://www.differencebetween.net/technology/difference-between-vb-and-vbscript/

https://www.safaribooksonline.com/library/view/vbscript-in-a/1565927206/ch01s03.html

> VBA stands for Visual Basic for Applications and so is the small "for applications" scripting brother of VB. VBA is indeed available in Excel, but also in the other office applications.
>
> With VB, one can create a stand-alone windows application, which is not possible with VBA.
>
> It is possible for developers however to "embed" VBA in their own applications, as a scripting language to automate those applications.
>
> **The minor differences**
>
> **Hosted vs. stand-alone**: In practical, terms, when most people say "VBA" they specifically mean "VBA when used in MS Office", and they say "VB6" to mean "VBA used in the last version of the standalone VBA compiler (i.e. Visual Studio 6)". The IDE and compiler bundled with MS Office is almost identical to Visual Studio 6, with the limitation that it does not allow compilation to stand-alone dll or exe files. This in turns means that classes defined in embedded VBA projects are not accessible from non-embedded COM consumers, because they cannot be registered.
>
> **Continued development**: Microsoft stopped producing a stand-alone VBA compiler with Visual Studio 6, as they switched to the .NET runtime as the platform of choice. However, the MS Office team continues to maintain VBA, and even released a new version (VBA7) with a new VM (now just called VBA7.dll) starting with MS Office 2010. The only major difference is that VBA7 has both a 32- and 64-bit version and has a few enhancements to handle the differences between the two, specifically with regards to external API invocations.

> Summary:
>
> 1. VB is an event driven programming language that was designed to make computer programming easier for programming beginners; VBScript is an active scripting language that uses COM to access elements of the environment in which it is running.
>
> 2. VB does not have the possibility of multiple assignment, but does contain a variable array base and strong integration with Windows; VBScript functions as a language which writes executable functions that are embedded in or included from HTML pages, and is known to create applications which run directly on a user’s computer if that computer is running Microsoft Windows.

Differences Between VBScript and VBA

VBScript is a subset of the Visual Basic for Applications language. There are several features that VB and VBA programmers have become accustomed to that are not present in VBScript. This does not lessen the usability of VBScript: it only serves to reinforce that VBScript is meant for scripting and not full-blown client/server application development or COM component development. Let's take a look at a few of the larger differences between VBScript and VBA:

VBScript is an untyped language.

Unlike Visual Basic and Visual Basic for Applications, in which the developer can define the data type of a variable in advance, all variables in VBScript are variants. There are subtypes to handle different types of data, and you can use these as you would the traditional data types in Visual Basic. For more information, see Chapter 3.

VBScript is not compiled.

Although we speak of VBScript code being " compiled" as it is downloaded, VBScript is nevertheless an interpreted language. That means that the code that you write is interpreted into machine language each time you run the script, which imposes a definite performance penalty. Visual Basic programmers who have worked with the language for a long time can remember when Visual Basic was also an interpreted language. Is this a big deal for VBScript? We don't believe so, since most scripting languages (JavaScript, Perl, Python, etc.) are interpreted rather than compiled. And code portions whose performance are particularly sensitive can be removed from the VBScript code and placed in a compiled COM component that the VBScript code instan-

VBA environments using the Object Browser, this is not the case with late-bound objects. Finally, the help facilities available for early-bound objects in VB and VBA (like Auto List Members and Auto Quick Info) are not available, making syntax errors more likely and ready access to good documentation all the more necessary.

VBScript does not support named arguments.

Most functions and procedures, VBA supports both positional and named arguments. For example, the VBA `MsgBox` function can be called using positional arguments as follows:

```
lResult = MsgBox("Delete this file?", _
          vbYesNo Or vbQuestion Or vbDefaultButton2, _
          "Confirm File Deletion")
```

A method call using named arguments takes the following form:

```
lResult = MsgBox(Prompt:="Delete this file?", _
          Title:="Confirm File Deletion", _
          Buttons:=vbYesNo Or vbQuestion Or vbDefaultButton2)
```

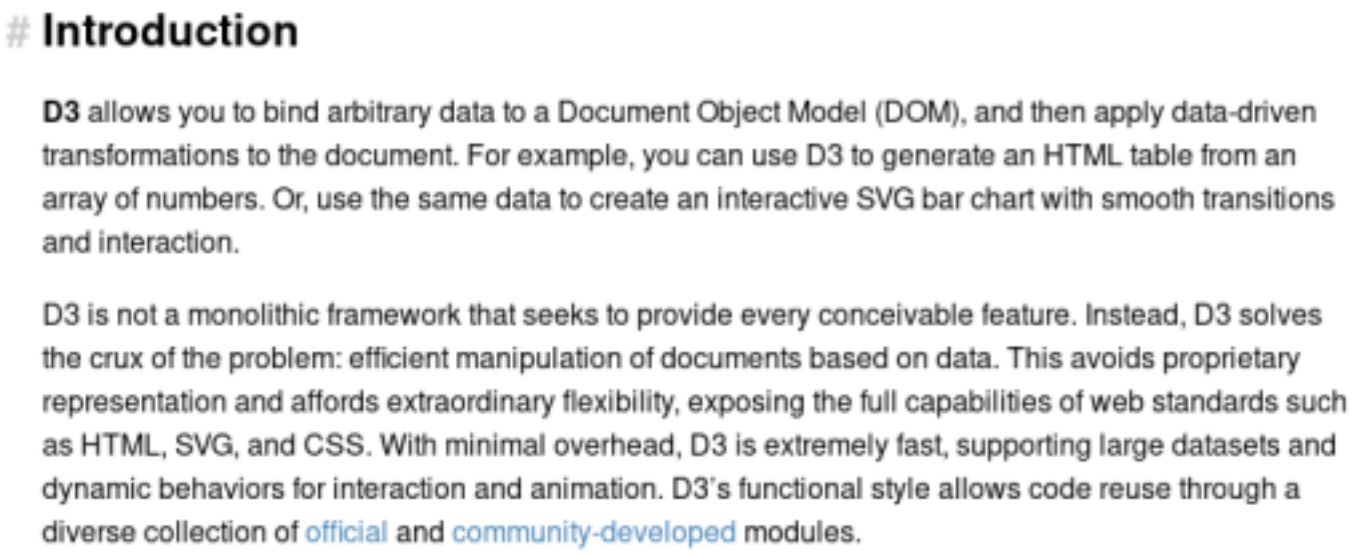
Note that while positional arguments must occur in a predefined sequence, named arguments need not. At least in our experience, more advanced programmers tend to prefer positional syntax, while more novice programmers tend to prefer named arguments.

VBScript does not have an IDE.

There is no integrated development environment for VBScript that parallels the IDE for Visual Basic and Visual Basic for Applications. Develpment tools are available for all of the environments in which VBScript is used, but all fall short of the power, simplicity, elegance, and ease of use of the VB/VBA IDE.Typically, web developers have had their own environments for writing their code. VBScript for the Web, whether it is client-side or server-side, is embedded inside of a `<SCRIPT>` tag. This allows web developers to continue to use their tool of choice even when using VBScript. And a wide array of tools for web script development are available. To mention just two from Microsoft, Microsoft Visual Interdev provides a visual interface for web application developers building ASP applications that perform data access. And the Microsoft Script Editor, which is included with Word 2000, provides something resembling (but that falls a little short of) an IDE for developing HTML pages with client-side and server-side scripts. Scripts for WSH can be created with the use of a simple text editor like Windows Notepad. Outlook comes with its rudimentary IDE (a glorified version of Notepad) for attaching code to Outlook forms.

D3js

https://d3js.org/



Introduction

D3 allows you to bind arbitrary data to a Document Object Model (DOM), and then apply data-driven transformations to the document. For example, you can use D3 to generate an HTML table from an array of numbers. Or, use the same data to create an interactive SVG bar chart with smooth transitions and interaction.

D3 is not a monolithic framework that seeks to provide every conceivable feature. Instead, D3 solves the crux of the problem: efficient manipulation of documents based on data. This avoids proprietary representation and affords extraordinary flexibility, exposing the full capabilities of web standards such as HTML, SVG, and CSS. With minimal overhead, D3 is extremely fast, supporting large datasets and dynamic behaviors for interaction and animation. D3's functional style allows code reuse through a diverse collection of official and community-developed modules.

Simple Example of D3js

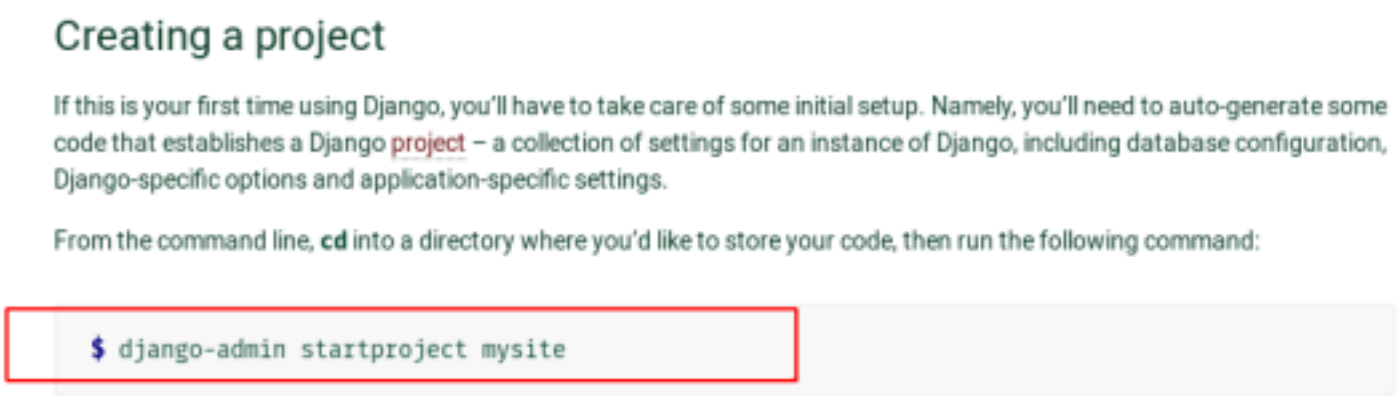
https://bost.ocks.org/mike/circles/

Django

https://www.djangoproject.com

Starting Tutorial

https://docs.djangoproject.com/en/2.0/intro/tutorial01/



Creating a project

If this is your first time using Django, you'll have to take care of some initial setup. Namely, you'll need to auto-generate some code that establishes a Django project – a collection of settings for an instance of Django, including database configuration, Django-specific options and application-specific settings.

From the command line, **cd** into a directory where you'd like to store your code, then run the following command:

```
$ django-admin startproject mysite
```

```
mysite/
    manage.py
    mysite/
        __init__.py
        settings.py
        urls.py
        wsgi.py
```

Running the Development Server

The development server

Let's verify your Django project works. Change into the outer **mysite** directory, if you haven't already, and run the following commands:

```
$ python manage.py runserver
```

You'll see the following output on the command line:

```
Performing system checks...

System check identified no issues (0 silenced).

You have unapplied migrations; your app may not work properly until they are applied.
Run 'python manage.py migrate' to apply them.

June 08, 2018 - 15:50:53
Django version 2.0, using settings 'mysite.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

To create your app, make sure you're in the same directory as **manage.py** and type this command:

```
$ python manage.py startapp polls
```

That'll create a directory **polls**, which is laid out like this:

```
polls/
    __init__.py
    admin.py
    apps.py
    migrations/
        __init__.py
    models.py
    tests.py
    views.py
```

MVC Framework

https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller

Model–view–controller

From Wikipedia, the free encyclopedia
(Redirected from Model-view-controller)

Model–view–controller is commonly used for developing software that divides an application into three interconnected parts. This is done to separate internal representations of information from the ways information is presented to and accepted from the user.[1][2] The MVC design pattern decouples these major components allowing for efficient code reuse and parallel development.

Traditionally used for desktop graphical user interfaces (GUIs), this architecture has become popular for designing web applications and even mobile, desktop and other clients.[3] Popular programming languages like Java, C♯, Ruby, PHP and others have popular MVC frameworks that are currently being used in web application development straight out of the box.

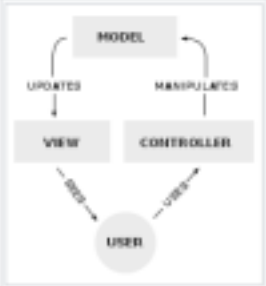


Diagram of interactions within the MVC pattern.

Contents [hide]

1 Descriptions
1.1 Components
1.2 Interactions
2 History

Datatypes

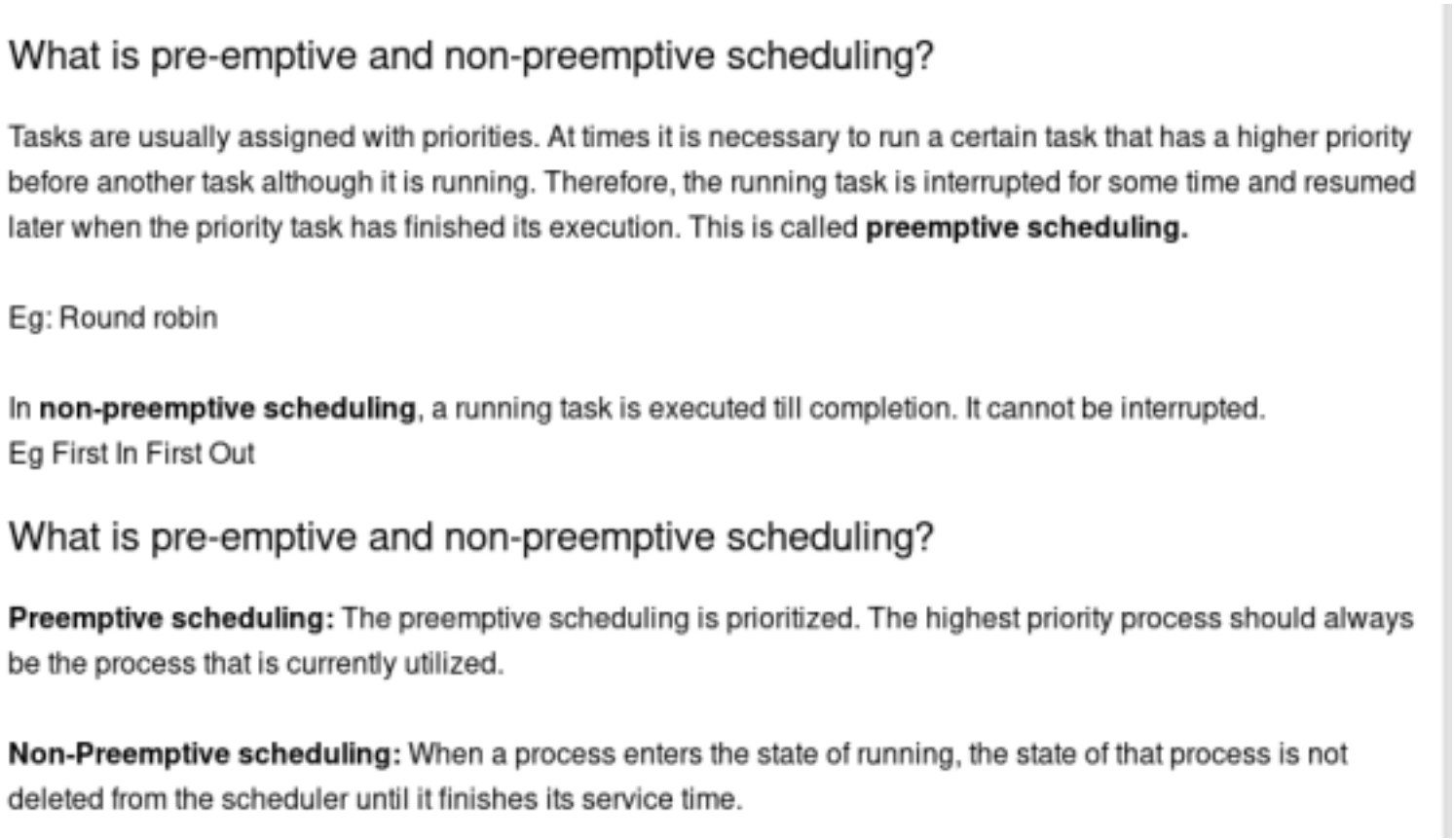
Type	Size in Bytes	Description
`Boolean`	2	Stores a value of `True` (0) or `False` (–1).
`Byte`	1	Contains a number in the range of 0–255.
`Currency`	8	Defines monetary values in the range of –922,337,203,685,477.5808 to 922,337,203,685,477.5807.
`Date`	8	Defines combination value that includes both date and time; although you certainly don't have to include both. Valid dates range from January 1, 100 to December 31, 9999.
`Decimal`	14	Defines a precise 28-digit number. You must use the `Variant` data type for this value. The number is in the range of ±79,228,162,514,264,337,593,543,950,335 with no decimal point and ±7.9228162514264337593543950335 with 28 places to the right of the decimal.
`Double`	8	Contains a real number in the range of –1.79769313486231 E308 to – 4.94065645841247 E-324 for negative values and 4.94065645841247 E-324 to 1.79769313486232 E308 for positive values.
`Integer`	2	Contains a number in the range of –32,768 to 32,767.
`Long`	4	Contains a number in the range of –2,147,483,648 to 2,147,483,647.
`Object`	4	Any object reference.
`Single`	4	Contains a real number in the range of –3.402823 E38 to –1.401298 E-45 for negative values and 1.401298 E-45 to 3.402823 E38 for positve values.
`String` (fixed length)	String length	Contains a set of characters from 1 to approximately 65,400 characters long.
`String` (variable length)	10 + string length	Contains a set of characters from 0 to approximately 2 billion characters long.
`Variant` (with characters)	22 + string length	Stores any valid non-numeric data type or data types larger than a `Double`.
`Variant` (with numbers)	16	Stores any valid numeric data type up to the size of a `Double`.

Scheduling

https://en.wikipedia.org/wiki/Scheduling_(computing)

https://en.wikipedia.org/wiki/Preemption_(computing)

https://en.wikipedia.org/wiki/Cooperative_multitasking



In computing, **preemption** is the act of temporarily interrupting a task being carried out by a computer system, without requiring its cooperation, and with the intention of resuming the task at a later time. Such changes of the executed task are known as context switches. It is normally carried out by a privileged task or part of the system known as a preemptive scheduler, which has the power to **preempt**, or interrupt, and later resume, other tasks in the system.

Preemptive Scheduling

1. Assign Priorities to Every Process
2. If a Higher Priority Process comes , then stop the Currently running Lower Priority Process
3. Wait for the Higher Priority Process to Complete
4. And Then start the Lower Priorioty Process

Cooperative Scheduling

1. Every Process has a fixed time and it cannot be interrupted
2. Once a Process is initiated all the other Process have to wait till the current running one stops

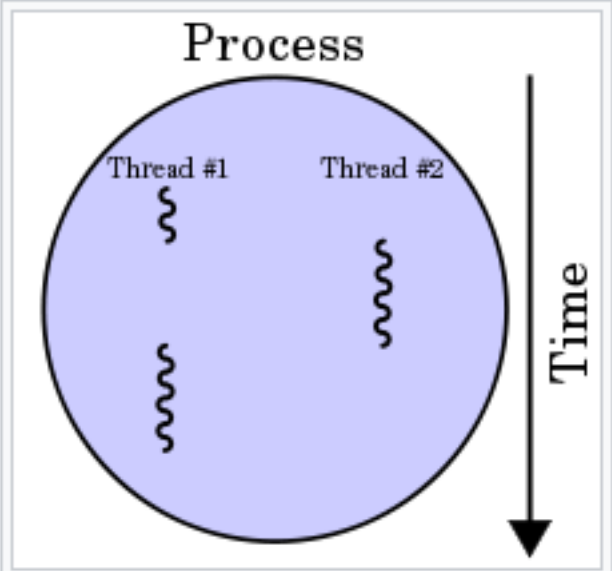
Cooperative multitasking

From Wikipedia, the free encyclopedia

Co-operative multitasking, also known as **non-preemptive multitasking**, is a style of computer multitasking in which the operating system never initiates a context switch from a running process to another process. Instead, processes voluntarily yield control periodically or when idle or logically blocked in order to enable multiple applications to be run concurrently. This type of multitasking is called "cooperative" because all programs must cooperate for the entire scheduling scheme to work. In this scheme, the process scheduler of an operating system is known as a **cooperative scheduler**, having its role reduced down to starting the processes and letting them return control back to it voluntarily.[1][2]

Although it is rarely used in modern larger systems, it is widely used in memory-constrained embedded systems and also, in specific applications such as CICS or the JES2 subsystem. Cooperative multitasking was the primary scheduling scheme for 16-bit applications employed by Microsoft Windows before Windows 95 and Windows NT (such as Windows 3.1x), and by the classic Mac OS. Windows 9x used non-preemptive multitasking for 16-bit legacy applications, as the PowerPC Versions of Mac OS X prior to Leopard used it for classic applications.[1] NetWare, which is a network-oriented operating system, used cooperative multitasking up to NetWare 6.5. Cooperative multitasking is still used on RISC OS systems.[3]

Thread, Process, Instances



A process with two threads of execution, running on one processor

Tasks

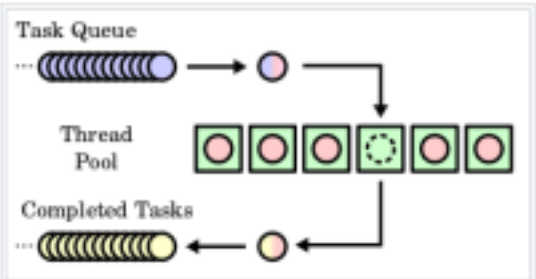
Task (computing)

From Wikipedia, the free encyclopedia

In computing, a **task** is a unit of execution or a unit of work. The term is ambiguous; precise alternative terms include *process*, light-weight process, *thread* (for execution), *step*, *request*, or *query* (for work). In the adjacent diagram, there are queues of incoming work to do and outgoing completed work, and a thread pool of threads to perform this work. Either the work units themselves or the threads that perform the work can be referred to as "tasks", and these can be referred to respectively as requests/responses/threads, incoming tasks/completed tasks/threads (as illustrated), or requests/responses/tasks.

Contents [hide]

1 Terminology
1.1 IBM terminology
1.2 Linux kernel
2 History
3 See also
4 References



A sample thread pool (green boxes) with task queues of waiting **tasks** (blue) and completed **tasks** (yellow), in the sense of task as "unit of work".

Threads vs. processes [edit]

Threads differ from traditional multitasking operating system processes in that:

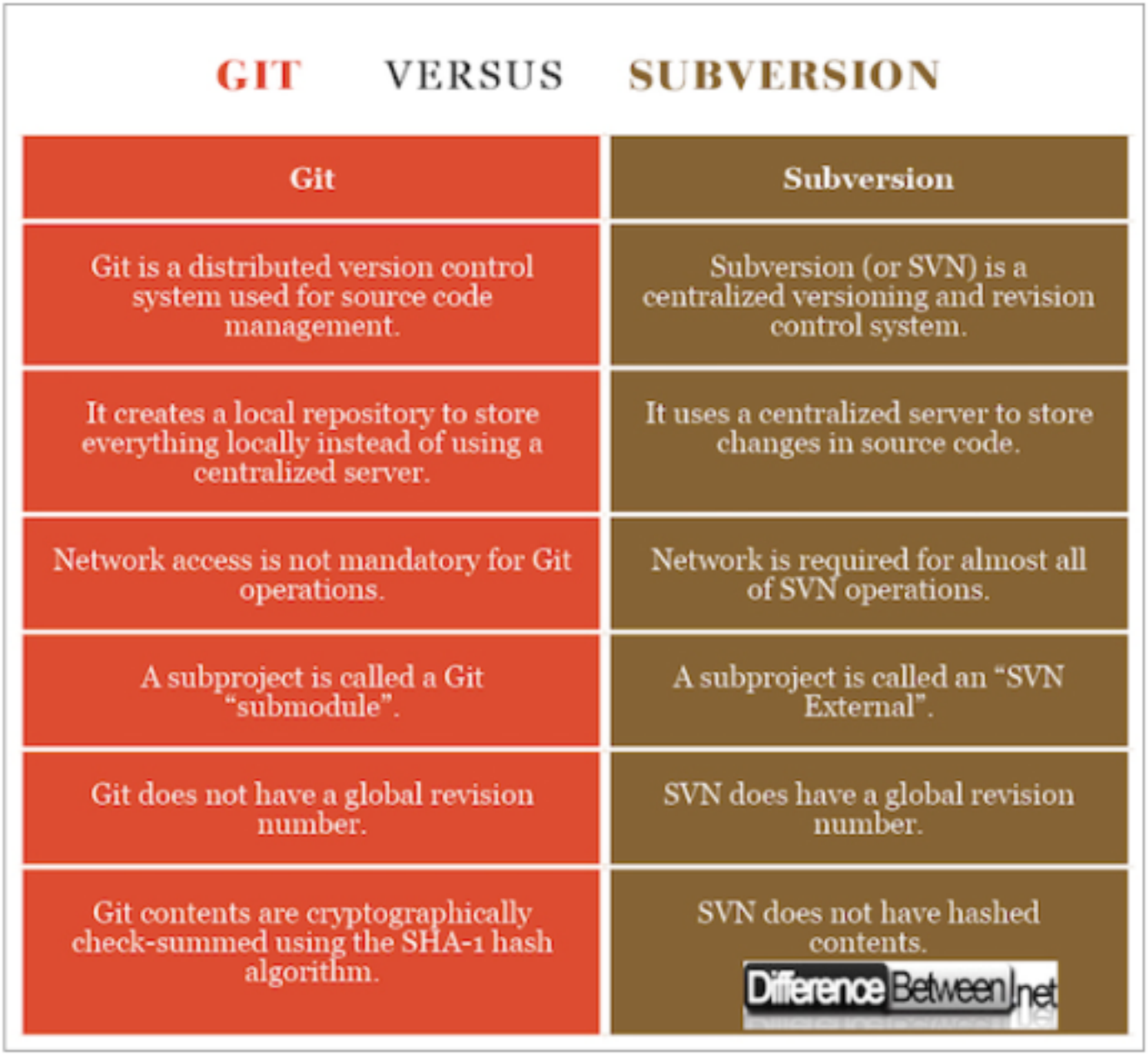
- processes are typically independent, while threads exist as subsets of a process
- processes carry considerably more state information than threads, whereas multiple threads within a process share process state as well as memory and other resources
- processes have separate address spaces, whereas threads share their address space
- processes interact only through system-provided inter-process communication mechanisms
- context switching between threads in the same process is typically faster than context switching between processes.

Systems such as Windows NT and OS/2 are said to have *cheap* threads and *expensive* processes; in other operating systems there is not so great a difference except the cost of an address space switch which on some architectures (notably x86) results in a translation lookaside buffer (TLB) flush.

Version Control

SVN vs Git

Version Control Comparison Chart



Differences Between Git and SVN

One of the most notable differences when switching to Git is its speed. Since the whole repository is stored locally on the developer's machine, he or she can work for days with a very poor internet connection. Creating branches is lightning fast due to Git's branch implementation. In Git, a branch is simply a reference to a commit, where the following commits will be attached. It doesn't contain even basic information like create date, user who created it or some kind of a message.

Since Git encourages the use of branches, we can't forget to give a shout-out to its merge capabilities. SVN before version 1.5 only did two-way merges that involved a change set applied to the current codebase, because it didn't store merge information. Git uses the history of the repository to identify the common base between the merged branches and only needs to merge from where they diverged — thereby completing a three-way merge. SVN is also improving and has supported three-way merging since 1.5. In the upcoming 1.9 version of SVN, it will also have better rename/move tracking of files, something that Git already does.

https://stackoverflow.com/questions/871/why-is-git-better-than-subversion

https://help.github.com/articles/what-are-the-differences-between-subversion-and-git/

https://markmcb.com/2008/10/18/3-reasons-to-switch-to-git-from-subversion/

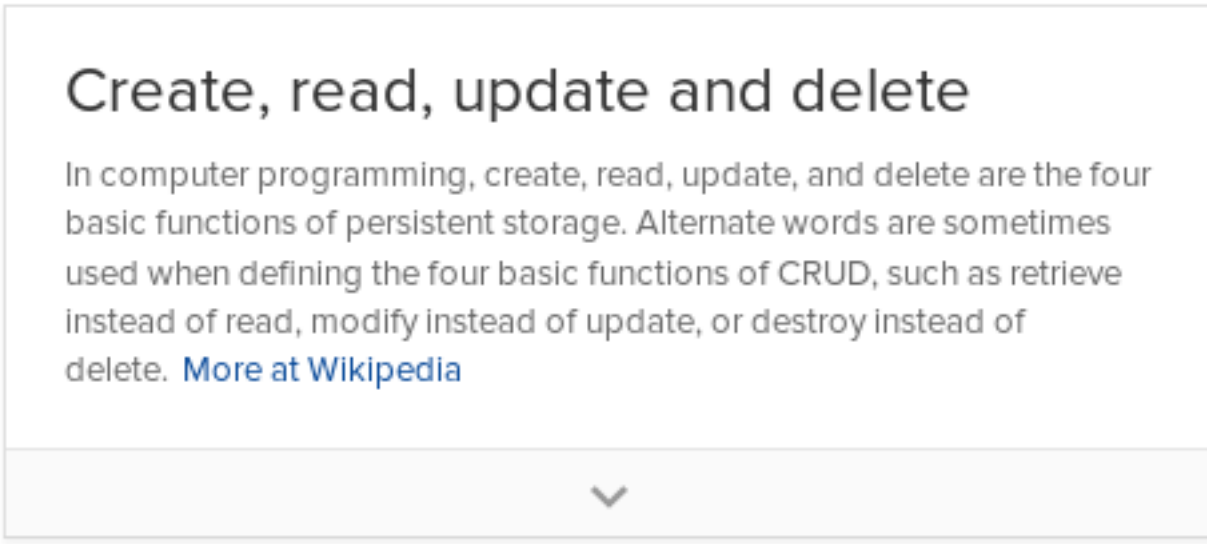
http://www.differencebetween.net/technology/difference-between-git-and-subversion/

Git - Decentralized, Has Full Local Copy, Detects only Changes

SVN - Centralized, Always need an Internet Connection, Has Full Copy and Detects all changes and pretty slow
in comparison with Git

Random

https://www.laptopmag.com/articles/install-unininstall-mac-software



Work

Brush Work Related Information

Domestic

International

2018

Germany

Zenuity - Dev Test Platform

Job Description

http://career.zenuity.com/jobs/97650-development-and-test-platforms

What you will do

- Develop and build state-of-the-art SiL or HiL solutions for Active Safety and Autonomous Driving ECUs and Sensors
- Maintain and drive development of these solutions based on testers feedback, requirements and latest technological progress
- Test implementation and -automation
- Integrate solutions in a continuous verification process
- Support and training for develoment teams

You will work with modern simulation environments, target hardware including vehicles for rapid development. You will work in agile teams which are responsible for all technical decisions regarding their product including the necessary synchronization with other regional or global teams.

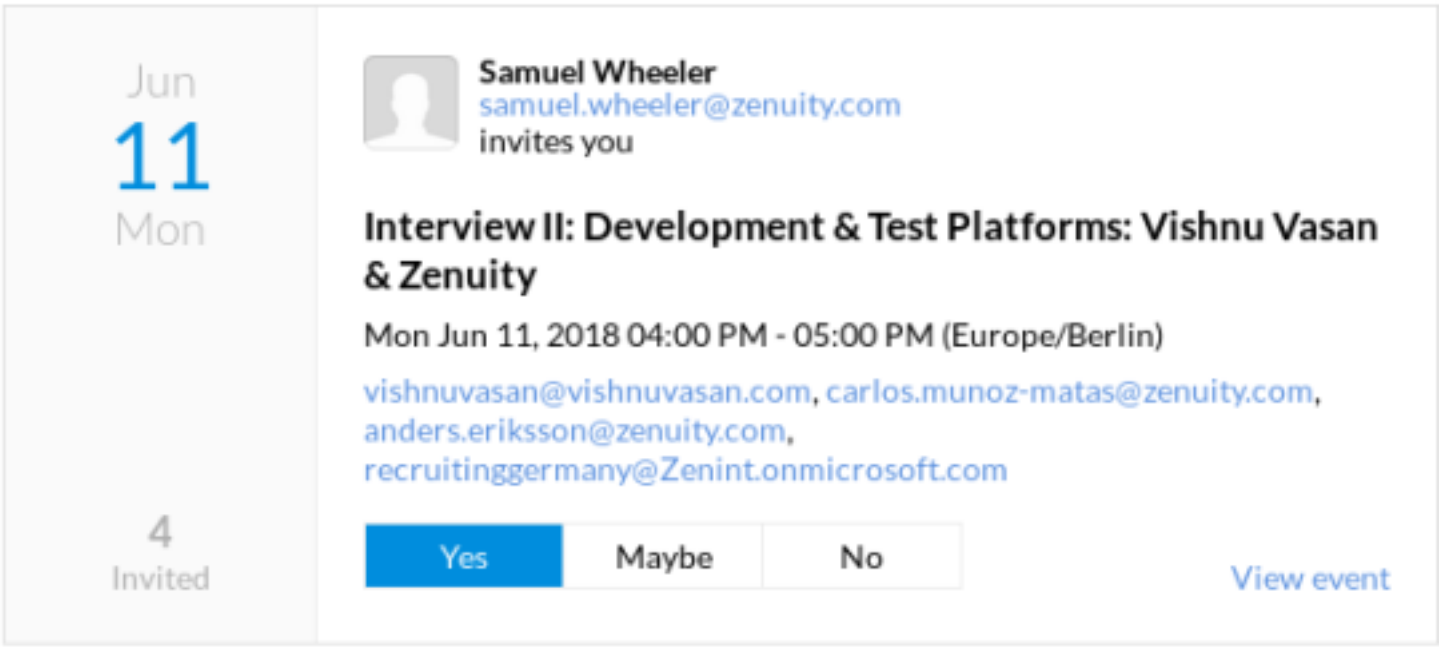
Your skills

Working in this area at Zenuity you have a strong background in building and maintaining test platforms. This includes some of the following experience/skills:

- automotive simulations
- Vehicle dynamics and sensor modeling
- familiarity with some of the following HiL/MiL/SiL tools: dSPACE, Vector, IPG, CarMaker, ASM, VTD Vires
- automotive communication protocols (CAN, Flexray, Automotive Ethernet)
- test implementation and automation
- Continuous integration
- programming experience (e.g. Python, Matlab/Simulink/Stateflow, C/C++)

Interview Schedules :

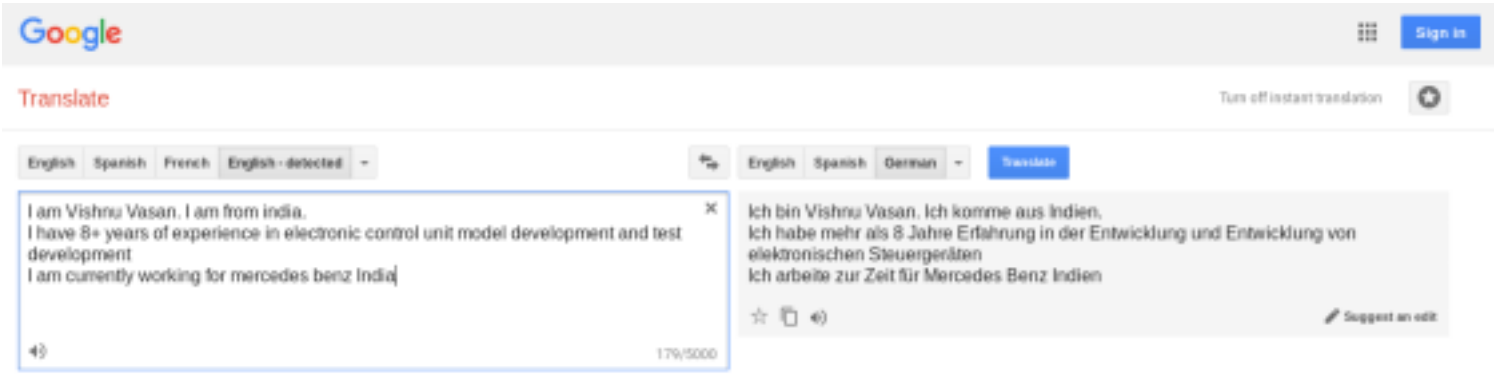
When?	What?	Status?	Comments
24th May 2018	1st Level Tech Recruiter Interview	Passed	
4th June 2018	2nd Level Online Assessment Interview	Passed	This round was total fun .Back to High School Mathematics and Stuff.But Managing the Time was the Difficult Part
11th June 2018	2nd Round In Depth Technical Interview with the Team	Waiting (Till 18th June). But Feel Busted	I would like to say that the expectation was different.They were pretty much concentrating on continous integration and linux experience which i dont have. so i feel that i am not the right match they are looking for. So it is highly likely that it is busted
14th June 2018	Official Result for 2nd Round Announced	Busted because of experience mismatch :)	But it was fun and Learning Experience



Overall Experience:

Takeaway:

Introduction



Questions

Why do you prefer coming to Europe?

1) True World Class Environment with Multi National Colleagues
2) Working with Some of the Best Minds. It gives me immense opportunity to Learn
3) Good Professional/Personal Life Balance

Why would you like to join our organization?

1) Working on Bleeding Edge Technologies and really being a part of the word “change”
2) Young and Energetic Org which means more Opportunities to Learn (OR)
3) One of the Best Org in the World, So to join hands to understand the best in class technologies
4) Agile, which means I have more opportunities to learn within a short cycle