

Software crítico para sistemas espaciales

2026/27

Master Universitario en Enxeñaría Aeroespacial

Arno Formella
Miguel Ramón González Castro
Manuel Pérez Cota

Departamento de Informática
Universidade de Vigo

26/27



- Software CRítico para Sistemas Espaciales
- acrónimo: (yo) SCRISE, (el centro) SOCRISE
- del lenguaje rumano
 - *scrise*
 - *los escritos*
 - *the writings*
- se dice: una imagen vale 1000 palabras
- yo pienso: una palabra vale 1000 imágenes,
- e incluso: una fórmula vale 1000 palabras!

- Arno Formella (unas 8 horas)
página web: <http://formella.webs.uvigo.es>
- Manuel Pérez Cota (unas 8 horas)
página web: <http://mpcota.webs.uvigo.gal>
- Miguel Ramón González Castro (unas 34 horas)
- quizá ponentes invitados

- los lunes de 17.30 a 19.00
- los miércoles de 16.30 a 19.00
- yo hago las primeras 2 semanas
- los profesores Pérez-Cota y González Castro siguen

- conocimiento adecuado de las tecnologías de información y las comunicaciones aeronáuticas
- comprender y aplicar los conocimientos, métodos y herramientas necesarias para desarrollar proyectos de ingeniería espacial
- concebir productos espaciales que correspondan a las necesidades de los agentes involucrados, definiendo funciones, conceptos y su arquitectura

según guía docente:

- participación **activa** en clase/conferencias (5%)
- entrega de **un trabajo** propio sobre algún tema específico del temario de la asignatura (40%)
- informe en modo de **presentación** sobre el trabajo realizado en el contexto de la asignatura (20%)
- (en caso de necesidad...) **examen** de preguntas (de desarrollo y cortas) (35%)

- Página web de la [European Cooperation for Space Standardization](#) (ECSS)
- Kshirasagar Naik y Priyadarshi Tripathy. *Software testing and quality assurance*. Wiley & Sons, ISBN 978-0-471-78911-6, 2008.
- Clay Breshears. *The Art of Concurrency*. O'Reilly, ISBN 978-0-596-52153-0, 2009.
- más material durante el curso

- página web pública acompañante:

<http://formella.webs.uvigo.es/doc/scrise26/index.html>

- Moovi:

<https://moovi.uvigo.gal/course/view.php?id=20373>

- sesiones en remoto:

<https://campusremotouvigo.gal/access/public/meeting/447269428>

[//campusremotouvigo.gal/access/public/meeting/447269428](https://campusremotouvigo.gal/access/public/meeting/447269428)
palabra clave estudiante (mirad en moovi)

- el material de estas traspas es muy densa
- haré una lista de material adicional disponible
- se intenta *no* usar muchas abreviaciones al contrario como suele ser normal en el sector
- no es un curso de programación, ni de diseño de software
- (para eso mejor contratar a un/a ingeniero/a informática

- software crítico es software que si falla pone en peligro
 - la vida o la salud de personas,
 - y/o un equipamiento caro o no re-emplazable,
 - y/o o el entorno con consecuencias graves al medio ambiente
- cuidado con los términos en inglés:
security versus safety – seguridad versus seguridad
- cuidado con ciberseguridad: usan el mismo término
software crítico pero más bien hacia *security*.

empezamos con algo aparentemente sencillo

- la vida de estudiante: se invitan a un@s colegas y se les quiere ofrecer algún pincho para comer
- receta de una tortilla española
- incluído el procedimiento de fabricación
- ¿que tiene que ver con espacial?
 - forma semejante a un ovni :-)
 - parece sencillo
 - pero fuente de muchos fallos posibles

empezamos con algo sencillo



<https://cookpad.com/es/recetas/16973981-tortilla-espanola-asi-la-preparo>



¿qué puede fallar? hay que ser creativo...

huevos contaminados por salmonelas; pérdida de energía (bombona vacía, tormenta y corto circuito); dar la vuelta no funciona (todo cae al suelo); falta de herramientas (sartén, plato, recipiente, etc.); cocción no está en su punto (fuera quemada dentro crudo); aceite está rancio; aceite no está a temperatura correcta (demasiado frío o caliente); receta incorrecta (con/sin cebolla); tamaño incorrecto (demasiado pequeño o grande); segunda fase con demasiado aceite; incorporación de demasiado aceite a los huevos (tortilla aceitosa); punto incorrecto de sal; incorrecto punto de cocción de las patatas y/o cebollas; echado azúcar en vez de sal; los huevos no se han batido; el aceite salpica toda la cocina; los invitados no vienen; hay alérgicos al huevo entre los invitados; no les gusta sin/con cebolla; los invitados son veganos; ya no hay tortilla para el último que viene a la fiesta; la tortilla no está lista cuando timbran; y muchas posibles incidencias más...



diseño de software: requisitos y especificaciones

sin entrar al principio en el tema complejo de diseño de software miramos dos aspectos importantes en este ámbito:

- requisitos

- describen el **qué**, es decir, que hay que hacer (en tal proyecto, para tal aplicación, para tal misión etc.)
- al final del proceso se debe **validar** que el producto cumple los requisitos impuestos

- especificaciones

- describen el **cómo**, es decir, como se tiene que hacer (tal proyecto, tal aplicación, tal misión etc.)
- al final del proceso se debe **verificar** que el producto cumple las especificaciones impuestas

a eso entramos más en adelante... pero como breve ejemplo: ordenar algo.

lista de eventos, digamos, con problemas...

(1962) Mariner 1, (1965) Gemini 3, (1965) Gemini 5, (1968) Apollo 8, (1969) Apollo 10, (1981) STS-1 (1st Shuttle), (1982) Viking-1, (1985) Therac-25, (1988) Phobos-1, (1988) Soyuz TM-5, (1991) Aries-Red Tigress I, (1991) Patriot Missile, (1992) F-22 Raptor, (1993) LH A320 accident Warsaw, (1994) Clementine Lunar Mission, (1994) Pegasus XL STEP-1, (1994) Pegasus HAPS, (1994) China Airlines Airbus A300, (1995) AA B-757 accident Cali, (1996) Ariane 5 Maiden Flight, (1997) Pathfinder, (1997) Delta III, (1997) Mars Polar Lander, (1999) Mars Climate Orbiter, (1999) Titan IV B Centaur for Milstar Satellite, (2000) Zenit 3SL, (2001) Pegasus XL/HyperX Launch Vehicle / X-43A, (2001) STS-108 til 110, (2002) Tim-Sort bug, (2003) Multidata Systems Radiation Machine, (2003) Soyuz TMA-1, (2003) North American Electric Power Grid,

lista de eventos, digamos, con problemas...

(2005) CryoSat-1, (2005) DART, (2006) Mars Global Surveyor, (2007) F22 First Deployment, (2008) STS-124, (2008) Quantas Flight 72, Airbus A330-303, (2008) B-2 Spirit - Guam crash, (2012) Red Wings Flight 9268 TU-204 (2015) Bloomberg terminal, (2015) Airbus A400M test flight (2015) SpaceX CRS-7 (2015) F-35 target detection, (2015) Starbucks POS, (2016) Schiaparelli test lander, ExoMars, (2016) Hitomi X-ray space telescope, (2017) SpaceX CRS-10, (2018)+(2019) Boeing 737 Max crash, (2019) Boeing Orbital Flight Test, (2019) Beresheet, (2020) Amazon Web Service (AWS) Kinesis, (2020) BD Alaris Infusion Pump, (2021) Global Facebook Outage, (2021) International Space Station, (2021) Volkswagen E³ 2.0, (2024) CrowdStrike, (2025) Cloudflare,...

(1981) STS-1 (1st Shuttle)



(1981) STS-1 (1st Shuttle)

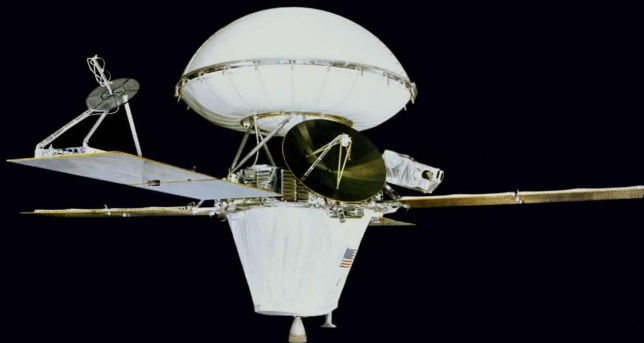
escenario: primer lanzamiento de un *space shuttle*

incidencia: 20 minutos antes del primer lanzamiento los relojes de los ordenadores estaban desincronizados como consecuencia de unas modificaciones del software uno y dos años antes, la probabilidad era de 1:67 que ocurriese el fallo

consecuencias: el lanzamiento se tenía que posponer dos días

remedio: básicamente se reinició el sistema, nada más (después de un análisis de las causas)

(1982) Viking-1



(1982) Viking-1



(1982) Viking-1

escenario: la nave espacial Viking I llegó con su vehículo de superficie y satélite de órbita a marte en el año 1976

incidencia: un comando para ajustar el proceso de carga de batería (en el año 1982) sobre-escribió una zona de memoria reservada para el apuntamiento de la antena

consecuencias: pérdida de comunicación con el sistema y terminación forzada de la misión

remedio: ninguno

(1991) Aries-Red Tigress I



(1991) Aries-Red Tigress I

escenario: el cohete militar Aries fue lanzado con una carga secreta de experimentos en la época de *Star Wars* de los años noventa

incidencia: unos 20 segundos después del lanzamiento, dado que el cohete se dirigió en dirección incorrecta, se destruyó, causa fue (dependiendo de la fuente) un fallo en un ordenador de control o la carga errónea del software de teleguiado (software de prueba en vez de software de vuelo)

consecuencias: cohete destruido y experimento fallido (se repitió unos dos meses más tarde)

remedio: cargar el software correcto y/o prevenir sistemas redundantes de control

(1991) Patriot Missile



(1991) Patriot Missile

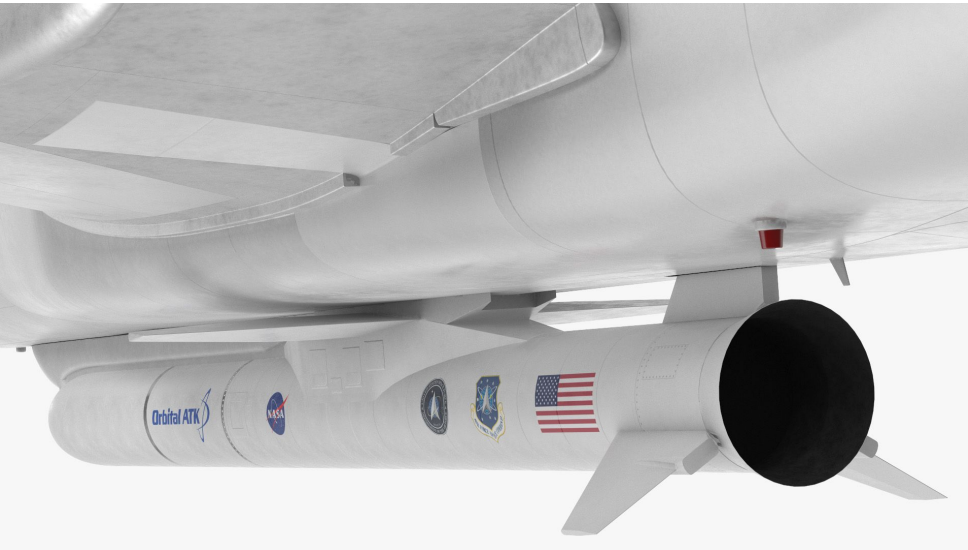
escenario: el cohete Patriot diseñado en los 70 intercepta otros objetos volando (o en tierra), se calcula el tiempo de abordó respecto al momento de inicio del sistema

incidencia: el código se basaba en 24-bits con bastante error de redondeo, después de las 100 horas en activo, el cohete tenía un error de desfase acumulado de 0.34 segundos que finalmente hizo imposible alcanzar el objeto previsto (Scud) y se destruyó un campamento propio

consecuencias: 28 muertos y unos 100 heridos

remedio: actualización de software en nuevos cohetes Patriot para evitar tal problema en el futuro

(1994) Pegasus XL STEP-1



(1994) Pegasus XL STEP-1

escenario: el cohete Pegasus XL STEP-1 (*space test experiments platform*) con varias etapas está diseñado para ser lanzado desde un avión y llevar cierta carga a una órbita baja

incidencia: haber usado un modelo aerodinámico no suficientemente preciso para el cohete más largo (XL), el control automático de trayectoria no pudo mantener el cohete en rumbo correcto y por intervención humana se destruyó el cohete después de unos 30 segundos de vuelo

consecuencias: pérdida de dos satélites del departamento de defensa

remedio: actualizar el modelo y el software para futuras misiones



ARIANE 5

FIRST FLIGHT FAILURE

V501 - JUNE 4, 1996



0:01 / 2:07 • Intro >



(1996) Ariane 5 Maiden Flight

escenario: el cohete Ariane 5 fue el sucesor de la Ariane 4 para llevar cargas útiles más grandes al espacio

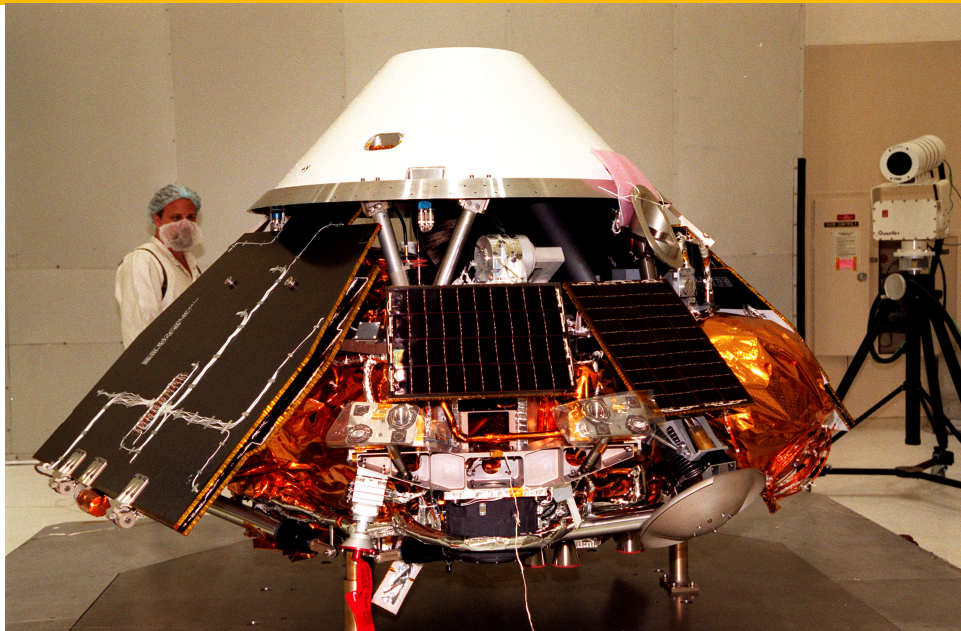
incidencia: en su vuelo inaugural el cohete fue a una trayectoria errónea y tanto el sistema de seguridad automático como humano daban orden de destrucción, un software de control de velocidad horizontal estaba copiado de la Ariane 4 y siguió en función ya despegado el cohete (innecesariamente), tal software, por el peso adicional de la versión 5, generó datos que no se pueden convertir a la precisión disponible (64-bit a 16-bit), tal error (no previsto) llegó al sistema duplicado pero idéntico (hardware y software) y se interpretó como desvío excesivo en vertical...

consecuencias: pérdida de la carga útil que consistía de 4 satélites Cluster para el estudio de la magnetosfera terrestre, unos 500 000 000 \$

remedio: reconstrucción de los 4 satélites Cluster y lanzamiento 4 años más tarde con un cohete Soyuz



(1999) Mars Polar Lander



(1999) Mars Polar Lander

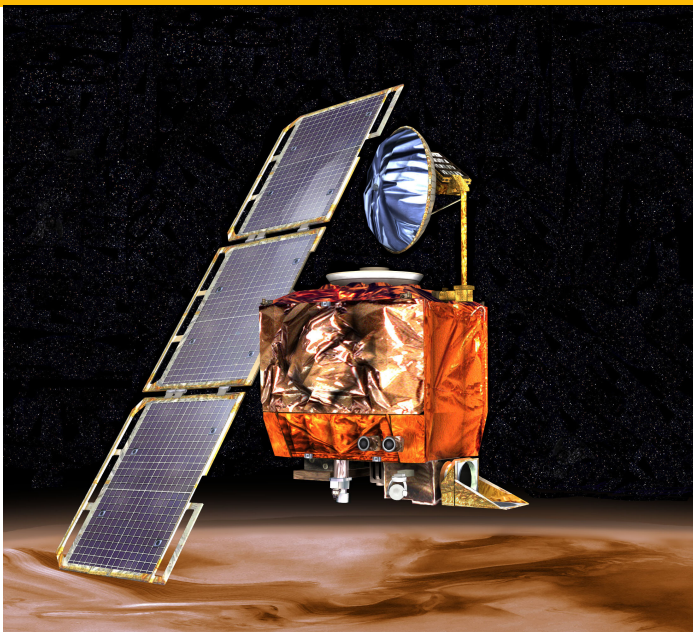
escenario: parte de la misión Mars Surveyor, el vehículo Mars Polar Lander fue lanzado para amartizar 9 meses después y estudiar la atmósfera de marte en esta región

incidencia: durante la bajada, 40m antes de la superficie una vibración de las patas causaba los sensores a mandar la señal *ya tocado*, faltaba una lógica correcta de interpretar todas las señales para interpretar la situación, se apagó el motor de frenado y el vehículo chocó desastrosamente

consecuencias: perdida de la misión

remedio: ninguno

(1999) Mars Climate Orbiter



(1999) Mars Climate Orbiter

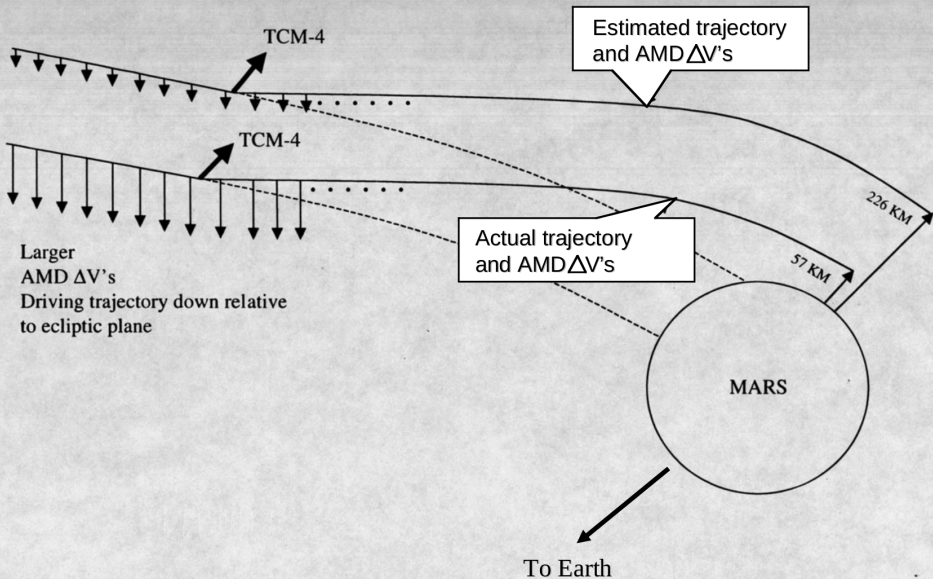
escenario: la segunda parte de la misión Mars Surveyer, que debería orbitar marte como observador del clima y estación repetidora para Mars Polar Lander

incidencia: el software de control de trayectoria desde la tierra a marte usaba por parte de las especificaciones de la NASA unidades métricas (y software de abordó), la implementación del software de tierra unidades imperiales, y por eso el vehículo se aproximó demasiado a marte y no pudo entrar en órbita, seguramente había colisión con marte en algún momento

consecuencias: pérdida de la misión

remedio: ninguno

(1999) Mars Climate Orbiter



(2002) Tim-Sort bug

Array Sorting Algorithms

Algorithm	Time Complexity			Space Complexity
	Best	Average	Worst	Worst
<u>Quicksort</u>	$\Omega(n \log(n))$	$\theta(n \log(n))$	$O(n^2)$	$O(\log(n))$
<u>Mergesort</u>	$\Omega(n \log(n))$	$\theta(n \log(n))$	$O(n \log(n))$	$O(n)$
<u>Timsort</u>	$\Omega(n)$	$\theta(n \log(n))$	$O(n \log(n))$	$O(n)$
<u>Heapsort</u>	$\Omega(n \log(n))$	$\theta(n \log(n))$	$O(n \log(n))$	$O(1)$
<u>Bubble Sort</u>	$\Omega(n)$	$\theta(n^2)$	$O(n^2)$	$O(1)$
<u>Insertion Sort</u>	$\Omega(n)$	$\theta(n^2)$	$O(n^2)$	$O(1)$
<u>Selection Sort</u>	$\Omega(n^2)$	$\theta(n^2)$	$O(n^2)$	$O(1)$
<u>Tree Sort</u>	$\Omega(n \log(n))$	$\theta(n \log(n))$	$O(n^2)$	$O(n)$
<u>Shell Sort</u>	$\Omega(n \log(n))$	$\theta(n(\log(n))^2)$	$O(n(\log(n))^2)$	$O(1)$
<u>Bucket Sort</u>	$\Omega(n+k)$	$\theta(n+k)$	$O(n^2)$	$O(n)$
<u>Radix Sort</u>	$\Omega(nk)$	$\theta(nk)$	$O(nk)$	$O(n+k)$
<u>Counting Sort</u>	$\Omega(n+k)$	$\theta(n+k)$	$O(n+k)$	$O(k)$
<u>Cubesort</u>	$\Omega(n)$	$\theta(n \log(n))$	$O(n \log(n))$	$O(n)$



(2002) Tim-Sort bug

- escenario:** algoritmo de ordenación que se adapta a la secuencia por ordenar y es muy rápido (complejidad en tiempo $\mathcal{O}(n \cdot (\log p + 1))$, p número de *sub-secuencias monótonas máximas*), uso en Python y Java
- incidencia:** no se ha analizado rigurosamente el algoritmo, los búferes intermedios pueden resultar demasiado pequeños (a partir de unos 60 millones de valores)
- consecuencias:** en uso durante 13 años, abre puertas de vulnerabilidades de *buffer overflow* o posible terminación inesperada de programas
- remedio:** hoy (a partir de Python 3.11) se usa un derivado de timsort, llamado *powersort*, con un análisis teórico completo y riguroso

(2003) North American Electric Power Grid



(2003) North American Electric Power Grid

escenario: el suministro de energía eléctrica está organizado en una gran rejilla compleja e interconectada, tanto respecto al control como respecto a la propia potencia

incidencia: una serie de incidencias menores (fallo de una línea, necesidad de compensar potencia reactiva, falta de predicción correcta por falta de datos de introducción manual, calor inusual en verano, problemas de comunicación y confianza etc.) la rejilla se apagó en cascada incluso sin advertir a los operarios por existir errores en los ordenadores y/o su software

consecuencias: 50 millones de personas sin electricidad incluyendo ciudades grandes, probablemente unos 100 muertos (por consecuencias indirectas)

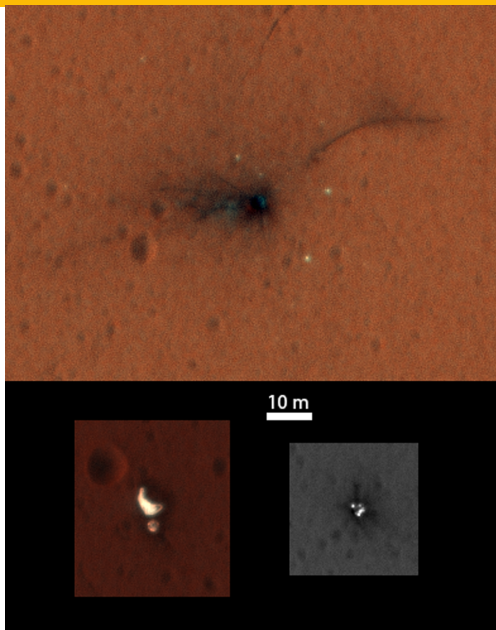
remedio: unos 46 recomendaciones como mejorar el control y la monitorización para evitar una repetición



(2016) Schiaparelli test lander, ExoMars



(2016) Schiaparelli test lander, ExoMars



(2016) Schiaparelli test lander, ExoMars

escenario: el vehículo Schiaparelli como parte de la misión *Trace Gas Orbiter* estaba previsto de realizar un amortizaje suave, a parte de realizar varias pruebas

incidencia: unos 3.7 km encima de la superficie con el despliegue de los paracaídas el vehículo se giraba muy deprisa, tal efecto causó que el sensor de velocidad llegó al límite y se interpretó como estar debajo de la superficie, se desató el paracaídas y se encendió brevemente el motor (no frenar más y subir un poco), luego se cayó en picado

consecuencias: pérdida de esta parte de la misión

remedio: ninguno (pero todo se pudiese haber resuelto antes con más simulaciones y una lógica que tenga en cuenta más datos de medición (disponibles) del vehículo)

(2018)+(2019) Boeing 737 Max crash



(2018)+(2019) Boeing 737 Max crash

escenario: la compañía quería, en vez de diseñar un nuevo avión, re-diseñar el avión existente para nuevos desafíos del mercado recortando así tiempos y costes, se colocaron motores más potentes en posiciones diferentes debajo de las alas, los problemas de control resultantes se compensaron con software actuando en el fondo

incidencia: los pilotos no estaban familiarizados con la actuación del software y el sistema automático con datos sensóricos de velocidad incorrectos (defecto en los sensores) daba comandos para bajar la nariz del avión hasta chocar contra el suelo

consecuencias: 346 muertos, dos aviones destrozados, pérdida de confianza y competitividad en el mercado

remedio: entrenamiento de los pilotos y refinamiento del software de control, cambio de política de seguridad



(2019) Beresheet



72

escenario: la nave Beresheet fue el primer intento de una empresa privada (israelita) de alunar un vehículo

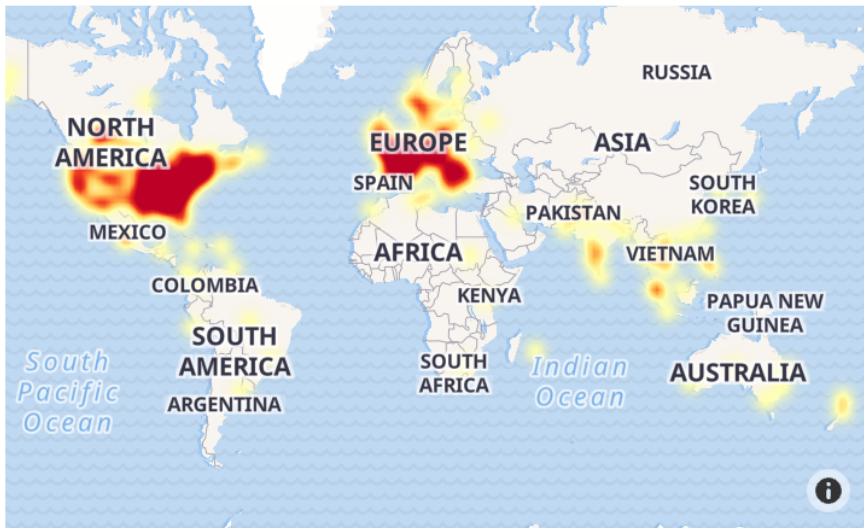
incidencia: había una cadena de pequeñas incidencias: fallo en electrónica (por radiación) que provocaba *reboots*, actualización de software solamente en RAM, cada reboot necesitaba tiempo (inicio más reconfiguración), reboot apaga motores, fallo en una IMU (*inertial measurement unit*), toma de decisión no arriesgar con solo una, intento de resucitar la fallida, no había tiempo para llevar a cabo todo para evitar la caída libre

consecuencias: finalmente la nave impactó forzosamente a la superficie de la luna con unos 135 m/s

remedio: ninguno

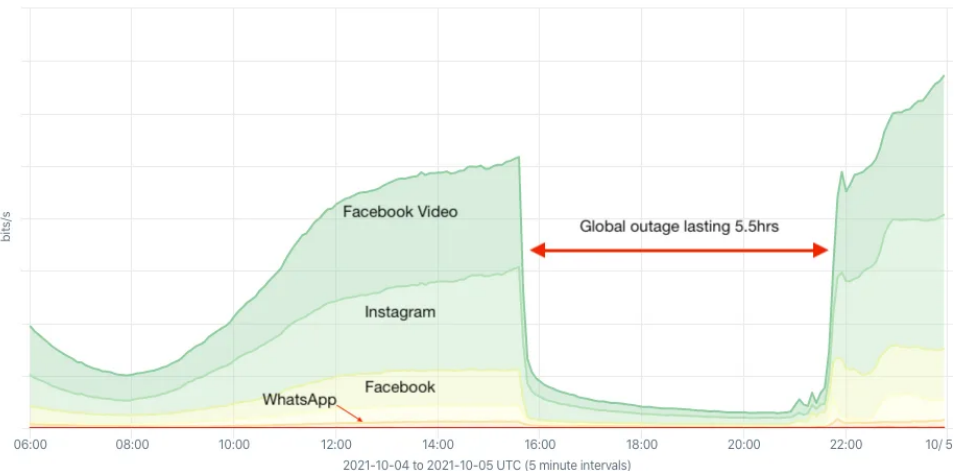
(2021) Global Facebook Outage

Facebook Outage Map Live



(2021) Global Facebook Outage

Top OTT Service by Average bits/s **Internet Traffic served by Facebook**
Oct 04, 2021 06:00 to Oct 05, 2021 00:00 (18h) **Global outage 4-Oct-2021**

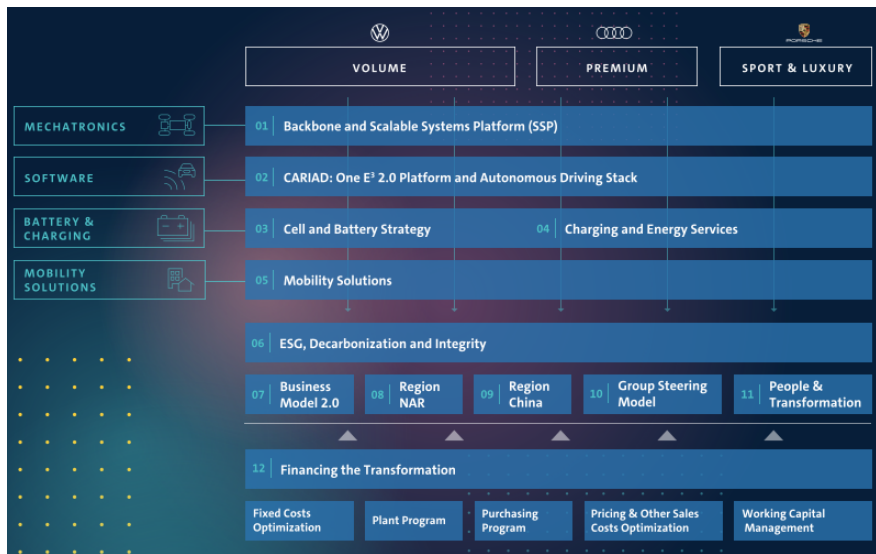


(2021) Global Facebook Outage

- escenario:** Facebook es una red social con usuarios distribuidos por todo el mundo (con más de 2 mil millones activos por día)
- incidencia:** por la ejecución de un comando erróneo en un centro de datos central se borró la información de enrutamiento en la red de otros centros de datos con el efecto de caída del sistema en muchas zonas a nivel mundial
- consecuencias:** la red no estaba utilizable para muchos millones de usuarios durante unas 5 horas, pérdida de valores en bolsa de Facebook, problemas generales en sistemas que dependen mucho del DNS, 60 000 000 \$ pérdidas en ganancias por anuncios, problemas de acceso a sistemas en remoto (que prolongó la toma de medidas)
- remedio:** resetear los servicios en los centros de datos una vez las personas autorizadas consiguieron acceso



(2021) Volkswagen E³ 2.0



(2021) Volkswagen E³ 2.0

CARIAD business model: Software is the key differentiator for the future, scale is key

VOLKSWAGEN

AKTIENGESELLSCHAFT

E³ 1.1

ID. family

E³ 1.2

PPE-platform
(Audi, Porsche)

E³ 2.0



SOFTWARE STACK
GROUPWIDE ROLLOUT

Artemis Trinity

FROM
2025

2020

2021

2022

2023

BREAK-EVEN
AFTER 2025

SOFTWARE
DEV IN
BRANDS

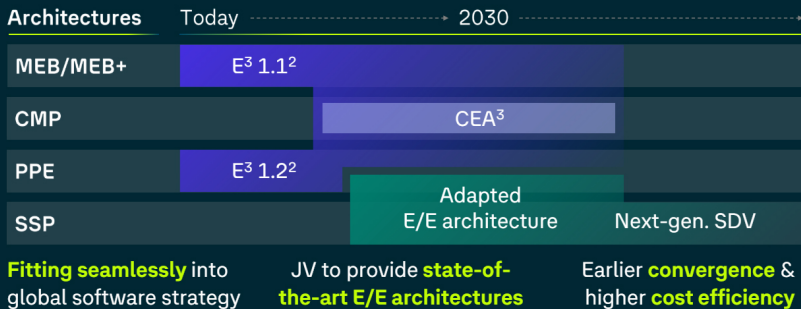
INVESTMENT PHASE

INCOME PHASE
THROUGH LICENSES

(2021) Volkswagen E³ 2.0

VOLKSWAGEN
GROUP

- ① Joint Venture¹ supports early convergence of vehicle software stack on state-of-the-art E/E architecture



1. Subject to the negotiation of definitive agreement | 2. Software stack by CARIAD | 3. CEA = China Electrical Architecture to be developed under Volkswagen Group's 'In China for China' strategy

7

(2021) Volkswagen E³ 2.0

- escenario:** la compañía se re-orienta para producir coches eléctricos para el mercado futuro con el abandono de motores de combustión (tarde o temprano)
- incidencia:** por problemas de gestión de riesgos y responsabilidades (en un monolito de empresa grande) no se llega a un desarrollo adecuado de tal proyecto software muy grande
- consecuencias:** hasta los años 30 no habrá un sistema operativo (o de control) íntegra para los coches eléctricos de la marca con las posibles consecuencias en el mercado
- remedio:** intento de buscar la solución fuera del país (inversión adicional de ya 5 000 000 000 de Euros) y recortes importantes en la empresa

(2024) CrowdStrike





Your PC ran into a problem and needs to restart. We're just collecting some error info, and then we'll restart for you.

85% complete

For more information about this issue and possible fixes, visit our website.

If you call a support person, give them this info:

Stop code: 0x00000000

(2024) CrowdStrike

escenario: un software antivirus de la empresa CrowdStrike es parte crítico de Windows-11 y plataforma Azure de la empresa Microsoft

incidencia: instalación en remoto de un fichero **erróneo** del sistema en muchos ordenadores comenzando en el este lejano (Australia), unos 8.5 millones sistemas entraban en estado de error (*Blue Screen of Death*, Pantalla Azul de la Muerte), bucle de rearranque infinito

consecuencias: coste estimado entre 6 000 000 000 \$ (empresas afectadas: bancos, aerolíneas, hospitales, emisoras) y 18 000 000 000 \$ (empresas causantes y costes indirectos), 5% de los vuelos cancelados

remedio: reiniciar en modo seguro a mano y arreglar el sistema de ficheros

tabla de eventos

	year	system	failure	reboot	recoverable	missing code	error location	unknown unknown	losses	casualties	Prokop
1	1962	Mariner 1 Mission – Atlas-Agena	Erroneous Output	No	No	Code/Logic	No	No	loss of vehicle		1
2	1965	Gemini 3	Erroneous Output	No	Yes	Code/Logic	Yes	No		short landing	2
3	1965	Gemini 5	Erroneous Output	No	No	Data	No	No		short landing	3
4	1968	Apollo 8	Erroneous Output	No	No	Command Input	No	No		human-in-loop	4
5	1969	Apollo 10	Erroneous Output	No	No	Data	No	No		human-in-loop	5
6	1981	STS-1	Fail Silent	Yes	Yes	Code/Logic	No	No		delay	6
7	1982	Viking-1	Erroneous Output	No	No	Command Input	No	No	loss of vehicle		7
8	1985	Therac-25	Erroneous Output	No	No	Code/Logic	No	No		4+(2)	8
9	1988	Phobos-1	Erroneous Output	No	No	Command Input	No	No	loss of vehicle		9
10	1988	Soyuz TM-5	Erroneous Output	No	No	Code/Logic	No	No		delay	10
11	1991	Aries - Red Tigris I	Erroneous Output	No	No	Sensor Input	No	No	loss of vehicle		11
12	1991	Patriot Missile	Erroneous Output	Yes	No	Code/Logic	No	No		28+(100)	12
13	1992	F-22 Raptor	Erroneous Output	No	Yes	Sensor Input	No	No	loss of vehicle	1	13
14	1993	LH A320 accident Warsaw	Fail Silent	No	No	Data	No	No			14
15	1994	Clementine Lunar Mission	Erroneous Output	No	No	Code/Logic	No	No	1/3-loss		15
16	1994	Pegasus XL STEP-1	Erroneous Output	No	Yes	Code/Logic	Yes	No	loss of vehicle		16
17	1994	Pegasus HAPS	Erroneous Output	No	Yes	Code/Logic	No	No	5/6-loss		17
18	1994	China Airlines Airbus A300	Erroneous Output	No	Yes	Code/Logic	No	No	loss of vehicle		18
19	1995	AA B-757 accident Cali,	Erroneous Output	No	Yes	Code/Logic	No	No	loss of vehicle		19
20	1996	Ariane 5 Maiden Flight	Erroneous Output	No	No	Code/Logic	No	No	loss of vehicle		20
21	1997	Pathfinder	Erroneous Output	No	No	Code/Logic	No	No		human-in-loop	21
22	1997	Delta III	Erroneous Output	No	Yes	Code/Logic	No	No	loss of vehicle		22
23	1997	Mars Polar Lander	Erroneous Output	No	Yes	Sensor Input	No	No	loss of vehicle		23
24	1999	Mars Climate Orbiter	Erroneous Output	No	No	Data	No	No	loss of vehicle		24
25	1999	Titan IV B Centaur (Milstar satellite)	Erroneous Output	No	No	Data	No	No	loss of vehicle		25
26	2000	Zenit 3SL	Erroneous Output	No	No	Code/Logic	No	No	loss of vehicle		26
27	2001	Pegasus XL/HyperX Launch Vehicle / X-43A	Erroneous Output	No	Yes	Code/Logic	Yes	No	loss of vehicle		27
28	2001	STS-108 through 110	Erroneous Output	No	No	Data	No	No		luck	28
29	2002	Tim-Sort bug	Fail Silent	No	Yes	Code/Logic	No	No		luck	29
30	2003	Multidata Systems Radiation Machine	Erroneous Output	No	No	Code/Logic	No	No		15+(many)	30
31	2003	Soyuz - TMA-1	Erroneous Output	No	No	Code/Logic	No	No		short landing	31
32	2003	North American Electric Power Grid	Fail Silent	Yes	No	Code/Logic	No	No	loss of revenues		32
33	2005	CryoSat-1	Erroneous Output	No	Yes	Code/Logic	No	No	loss of vehicle		33
34	2005	DART (Demonstration of Autonomous Rendezvous Technology)	Erroneous Output	No	No	Code/Logic	No	No	loss of vehicle		34
35	2006	Mars Global Surveyor (MGS)	Erroneous Output	No	No	Code/Logic	No	No	loss of vehicle		35
36	2007	F22 First Deployment	Fail Silent	No	Yes	Code/Logic	No	No		human-in-loop	36
37	2008	STS-124	Erroneous Output	No	Yes	Sensor Input	No	No		delay	37
38	2008	Quantas Flight 72, Airbus A330-303	Erroneous Output	No	Yes	Sensor Input	Yes	No		0+(119)	38
39	2008	B-2 Spirit - Guam crash	Erroneous Output	No	Yes	Sensor Input	Yes	No	loss of vehicle		39
40	2012	Red Wings Flight 9268 TU-204 crash	Erroneous Output	No	Yes	Code/Logic	Yes	No	loss of vehicle	5+(3)	40
41	2015	Bloomberg terminal	Fail Silent	No	No	Data	No	No	loss of revenues		41
42	2015	Airbus A400M test flight	Erroneous Output	No	No	Data	No	No	loss of vehicle	4	42
43	2015	SpaceX CRS-7	Erroneous Output	No	Yes	Code/Logic	No	No	loss of vehicle		43
44	2015	F-35 target detection	Erroneous Output	No	Yes	Code/Logic	No	No		delay	44
45	2015	Starbucks POS	Fail Silent	No	No	Data	No	No		human-in-loop	45
46	2016	Schiaparelli	Erroneous Output	No	Yes	Code/Logic	No	No	loss of vehicle		46
47	2016	Hitomi X-ray space telescope	Erroneous Output	No	No	Code/Logic	No	No	loss of vehicle		47
48	2017	SpaceX CRS-10	Erroneous Output	No	No	Data	No	No		delay	48
49	2018	737 Max crash (2019 again)	Erroneous Output	No	Yes	Sensor Input	Yes	No	loss of vehicle	346	49
50	2019	Boeing Orbital Flight Test (OFT)	Erroneous Output	No	No	Code/Logic	No	No		delay	50
51	2019	Bereshheet	Fail Silent	No	No	Code/Logic	No	No	loss of vehicle	human-in-loop	51
52	2020	Amazon Web Service (AWS) Kinesis	Fail Silent	No	Yes	Code/Logic	No	No	loss of revenues	human-in-loop	52
53	2020	BD AlarisTM Infusion Pump	Erroneous Output	No	No	Code/Logic	No	No		1+(55)	53
54	2021	Global Facebook Outage	Fail Silent	Yes	No	Command Input	No	No	loss of revenues	human-in-loop	54
55	2021	ISS	Erroneous Output	No	No	Code/Logic	No	No		human-in-loop	55
56	2021	Volkswagen E 2.0	Erroneous Output	No	Yes	Code/Logic	No	No	loss of revenues		56
57	2024	CrowdStrike	Fail Silent	No	No	Data	No	No	loss of revenues	human-in-loop	57



- os invito a **buscar y mirar** la información sobre los eventos presentados y los eventos no-presentados (y seguro que no pesqué todo :-)
- es interesante (y hasta divertido) observar todo lo que **puede fallar**
- en caso de errores en software se puede pensar en los responsables y sus emociones una vez haber **analizado la causa** del incidente no-deseado

para simplificar, se pueden categorizar las incidencias en principalmente 5 grupos:

- salida de datos incorrectos o caída del sistema
- recuperación posible con reinicio
- falta de código
- localización: código/lógica, datos, comandos, sensores
- incógnitas desconocidas

analizamos los 57 eventos respecto a estas categorías, más tarde retomamos ciertos aspectos

salida de datos incorrectos o caída del sistema

	#incidencias	porcentaje
salidas incorrectas	46	81 %
caída del sistema	11	19 %

cual de las dos posibilidades es peor depende de cada caso y si el *humano en el bucle* puede actuar respecto a la información que recibe

	#incidencias	porcentaje
de todos los eventos	4 de 57	7 %
salidas incorrectas	1 de 46	2 %
caída del sistema	3 de 11	27 %

el reinicio de sistema no es *la* solución, sino ayuda en ciertos casos (donde el fallo es transitorio, o probabilístico, o aislable en el rearranque)

remediable con más código

	#incidencias	porcentaje
de todos los eventos	23 de 57	40 %
salidas incorrectas	19 de 46	41 %
caída del sistema	4 de 11	36 %

mejorar el software promete menos incidencias con un alto porcentaje

localización del error en la arquitectura del software

localización	#incidencias	porcentaje
código/lógica/algoritmo	35	62 %
malconfiguración de datos	11	19 %
entrada desde sensores	7	12 %
error del operador/procedimiento	4	7 %

en grueso: la gran mayoría de problemas se pueden intentar resolver con mejor software y una cuidadosa configuración

ojo: los datos están **sesgados** ya que se trata de un ámbito muy especial con operadores bien entrenados y atentos (por eso el ratio del fallo humano suele ser pequeño), y con un hardware (y software) muy comprobado y/o redundante

incógnitas desconocidas

incidencias de las cuales no se sospechaba durante el desarrollo del sistema:

	#incidencias	porcentaje
incógnitas desconocidas	9	16 %

las incógnitas desconocidas (que quizá con más tiempo y más análisis se podrían prever) son causantes de una séptima parte de las incidencias

es un argumento adicional para incluir la persona responsable de la informática temprano en el proyecto para obtener un punto de vista más desde el inicio