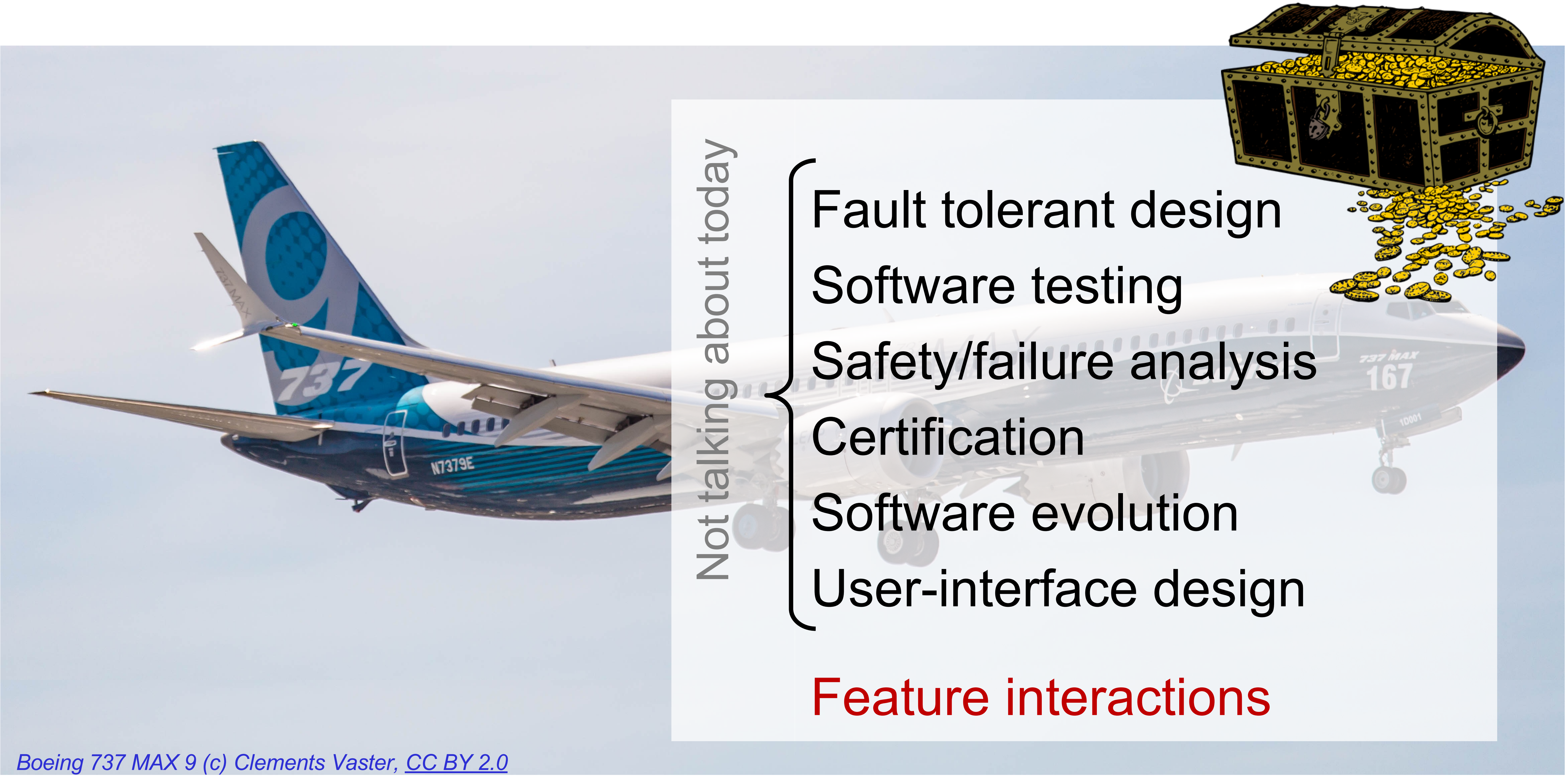
Feature-Oriented Requirements: The Good, the Bad, and the Ugly

Jo Atlee • CRSC • July 2021

UNIVERSITY OF WATERLOO | DAVID R. CHERITON SCHOOL OF COMPUTER SCIENCE

WATFORM
Waterloo Formal Methods

software in the Boeing 737 MAX 8



Not talking about today

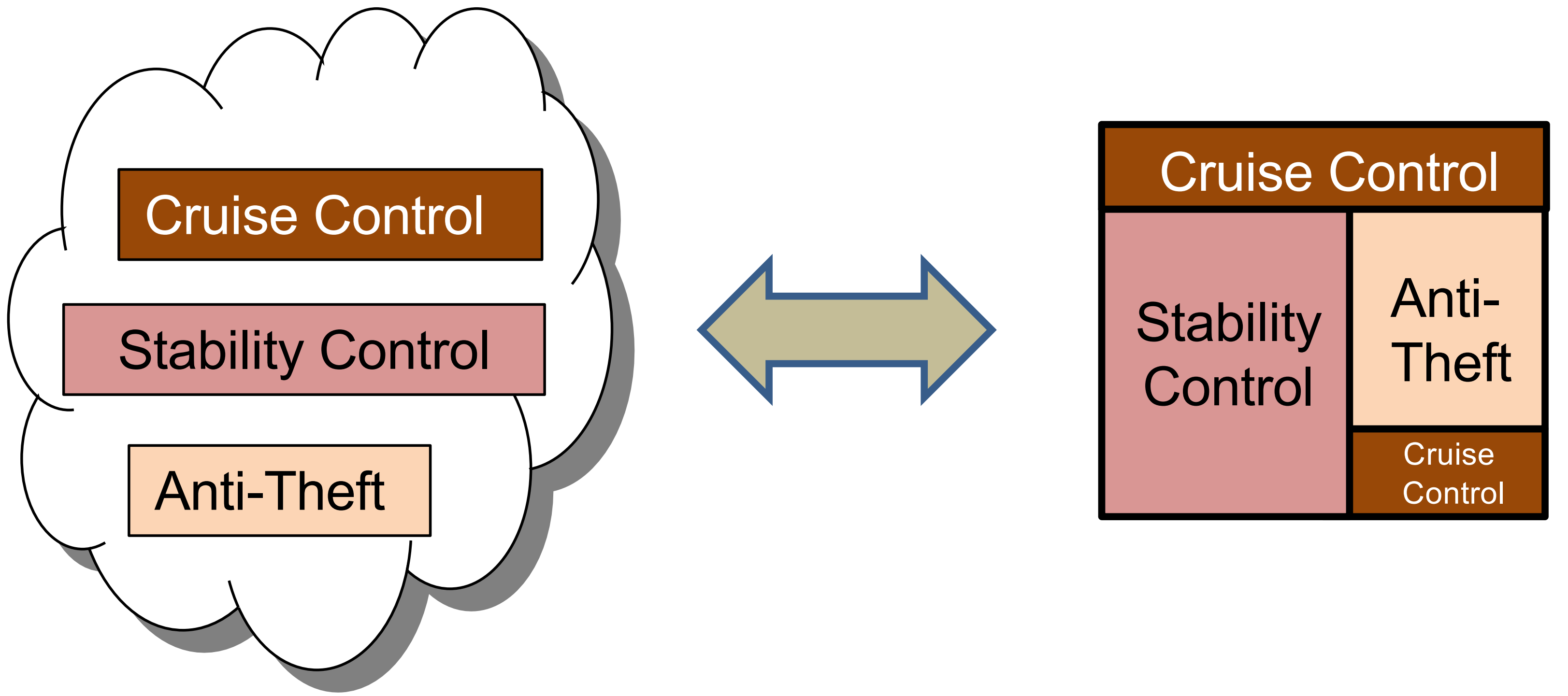
- Fault tolerant design
- Software testing
- Safety/failure analysis
- Certification
- Software evolution
- User-interface design

Feature interactions

some preliminaries

feature-oriented software

feature: a unit of added-value



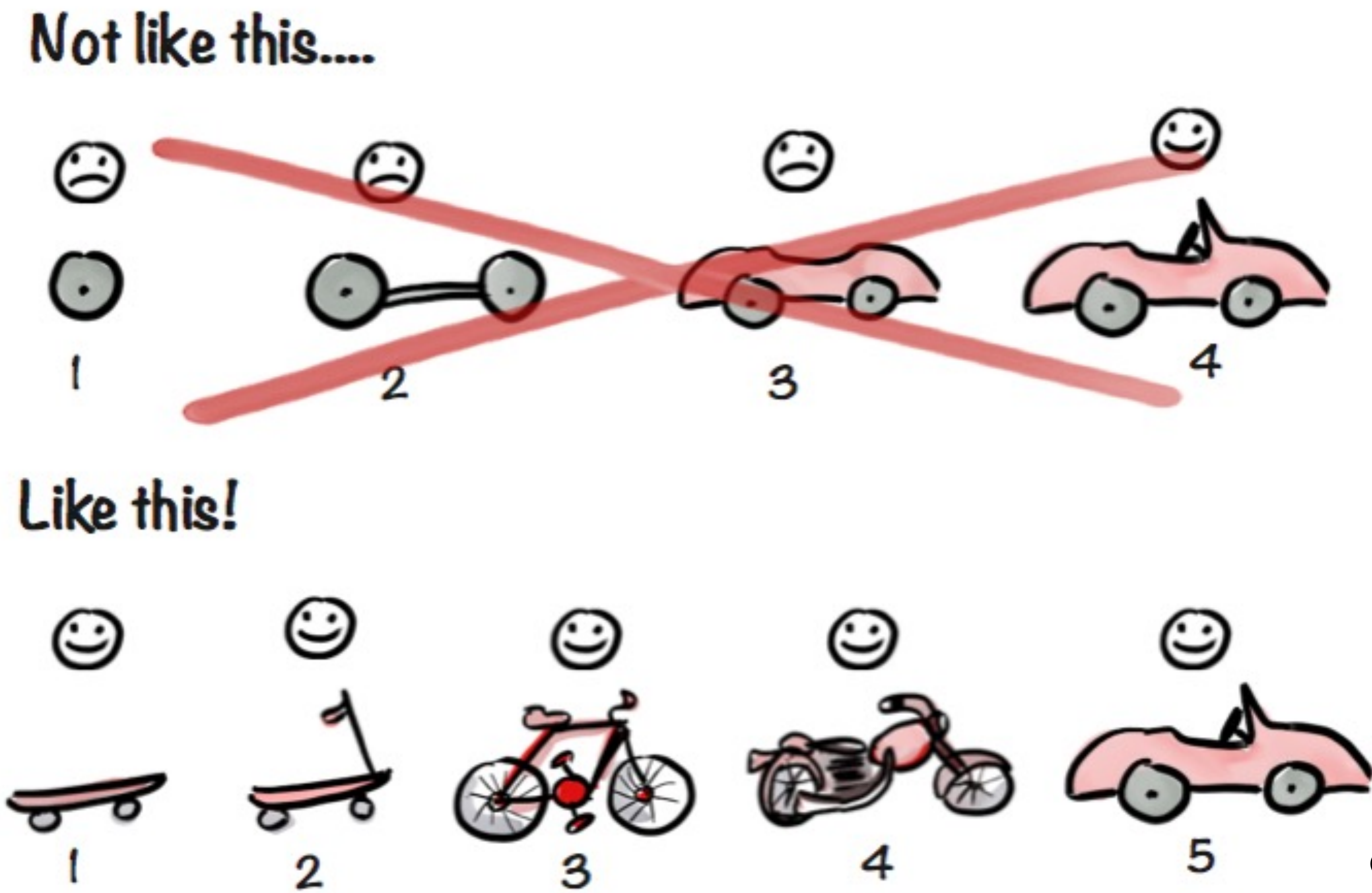
stakeholders' mental model of system

feature-oriented software system

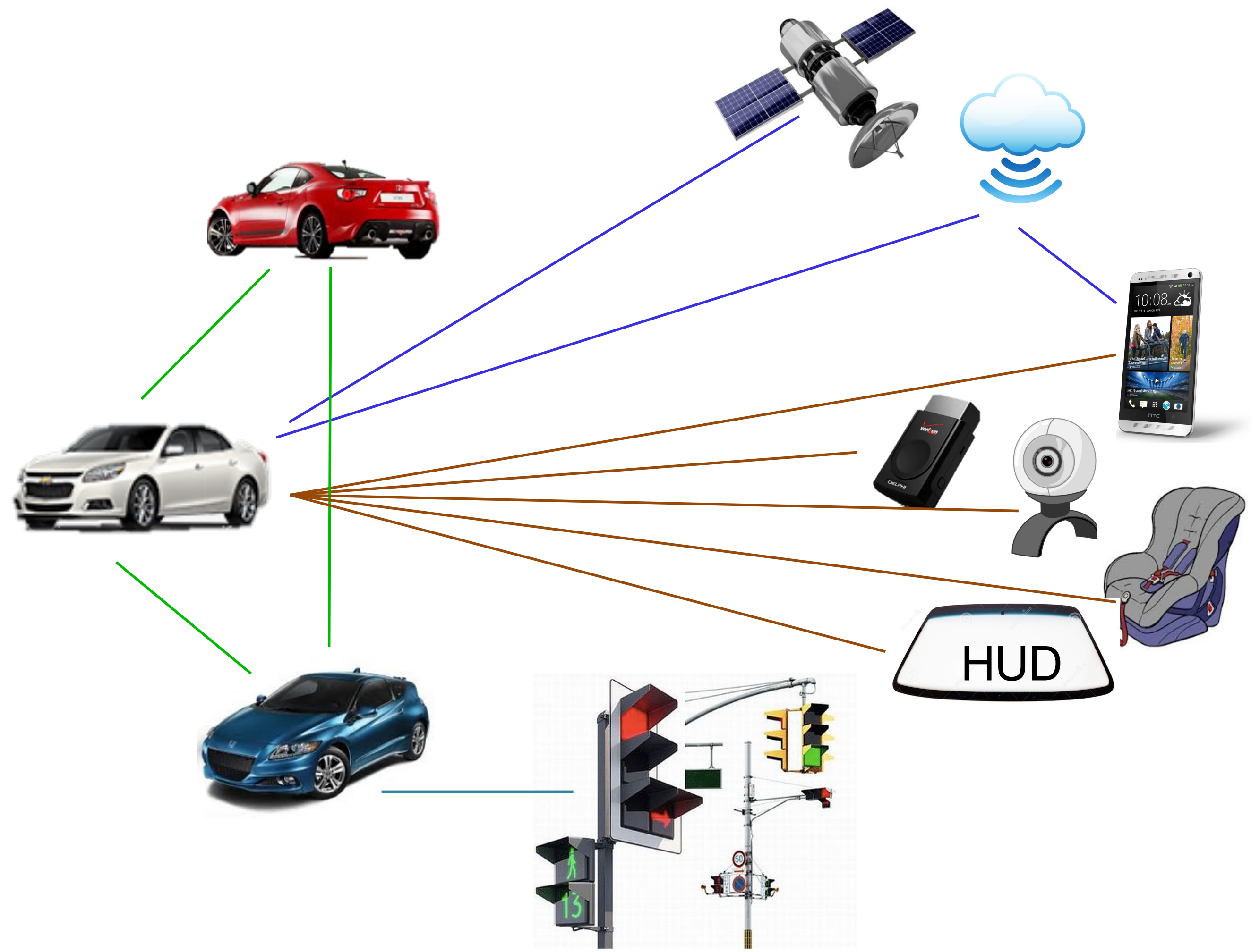
features
comparison shopping

Feature	Adobe Reader X	Acrobat X Standard	Acrobat X Pro	Acrobat X Suite
Read, print, and share PDF files				
View and print PDF files	•	•	•	•
More securely open PDF files in a sandboxed environment	•	•	•	•
Optimize your PDF viewing experience with Reading Mode	•	•	•	•
Store and share documents and forms using services at Acrobat.com[1]	•	•	•	•
Convert to PDF				
Create PDF files from any application that prints		•	•	•
Convert Microsoft Word, Excel, PowerPoint, Publisher, and Access files to PDF with one-button ease[2]		•	•	•
Scan paper documents into PDF and automatically recognize text with improved optical character recognition (OCR)		•	•	•
Capture web pages as interactive PDF files for review and archiving from Microsoft Internet Explorer and Firefox with one-button ease[2]		•	•	•
Archive emails or email folders from Microsoft Outlook or IBM® Lotus Notes with one-button ease[2]		•	•	•
Create PDF files from the clipboard, including text and images copied from external applications		•	•	•
Convert Autodesk® AutoCAD®, Microsoft Visio, and Microsoft Project files to PDF with one-button ease[2]			•	•
Export and edit PDF files				
Save PDF files as Microsoft Word documents and Excel spreadsheets, retaining the layout, fonts, formatting, and tables		•	•	•
Quickly and easily edit PDF files by making simple changes to text		•	•	•
Insert, extract, replace, delete, rotate, or reorder pages in a PDF file		•	•	•
Split large PDF files into multiple files based on maximum file size, maximum pages per file, or bookmarks		•	•	•
Add rich media to PDF files				
Insert audio, Adobe Flash® Player compatible video, and interactive media for direct playback in Acrobat and Adobe Reader®[2]			•	•
Convert a wide variety of video formats for smooth playback in PDF with Adobe Media Encoder				•
Edit and enhance photos to add to your PDF communications with Adobe Photoshop® CS5, the industry standard for image editing				•
Quickly transform static PowerPoint slides into compelling, interactive PDF presentations with Adobe Presenter				•
Rapidly combine audio, video, screen recordings, slides, and more into a rich media experience with Adobe Captivate®				•

features
incremental development



features
third-party functionality



feature interactions

feature interaction: features that behave well when executed in isolation, but behave *in different, expected, or undesired ways* when they execute together

feature interactions manifest themselves as

- conflicting actions
- nondeterminism
- resource contention
- performance degradation
- violated global correctness property
- inhibited behaviours
- emergent behaviours

hybrid brakes ⊕ anti-lock braking

U.S. NHTSA, (https://www.nhtsa.gov/vehicle/2010/TOYOTA/PRIUS/4%252520DR/FWD#investigations)

2010 Toyota Prius

hybrid brake system

- (normal) hydraulic brake system
- regenerative braking system
  - converts loss of vehicle momentum into electrical energy
  - stored in on-board batteries

anti-lock brake system (ABS)

- maintains stability, steerability during panic braking

interaction

- braking force after ABS actuation is reduced
- vehicle stopping distance is increased
- 62 reported crashes, 12 injuries

cruise control ⊕ traction control

cruise control
- vehicle set to maintain driver-specified speed

traction control
- brake fluid applied when wheels slip

interaction
- engine power is increased (to maintain speed)
- driver senses “sudden acceleration”
  - vehicle becomes difficult to control

resolution
- advise drivers not to use cruise control on slippery roads

good interactions

not all interactions are bad!

intended interactions

- advanced cruise control extends basic cruise control
- prohibit navigation overrides navigation
- prohibit-navigation override overrides prohibit-navigation

(planned) resolutions to conflicts

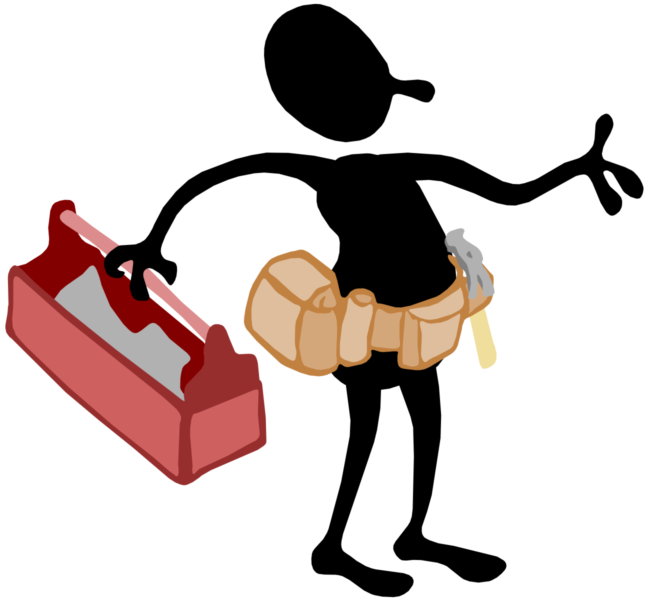
- brake override overrides (acceleration ⊕ braking)

unintended but harmless interactions

- call screening prevents activation of caller id

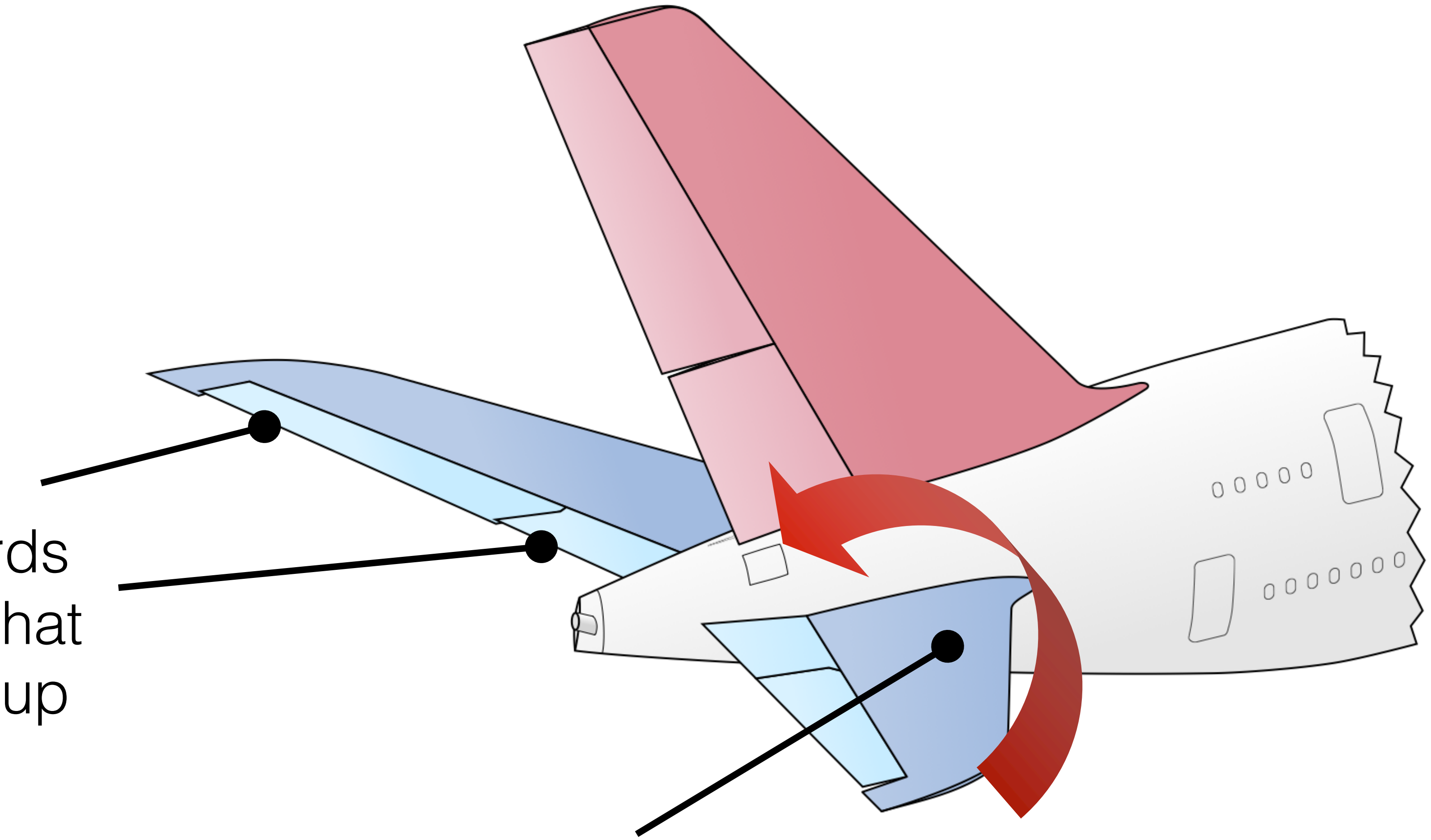
all interactions require work

- **verify *intended* interactions**
- **detect un*expected* interactions**
- **analyze them for un*desired* interactions**
- **fix undesired interactions**
  - faulty feature
  - disallow feature combination
  - resolve interaction
- **test the fixes**



bad interactions

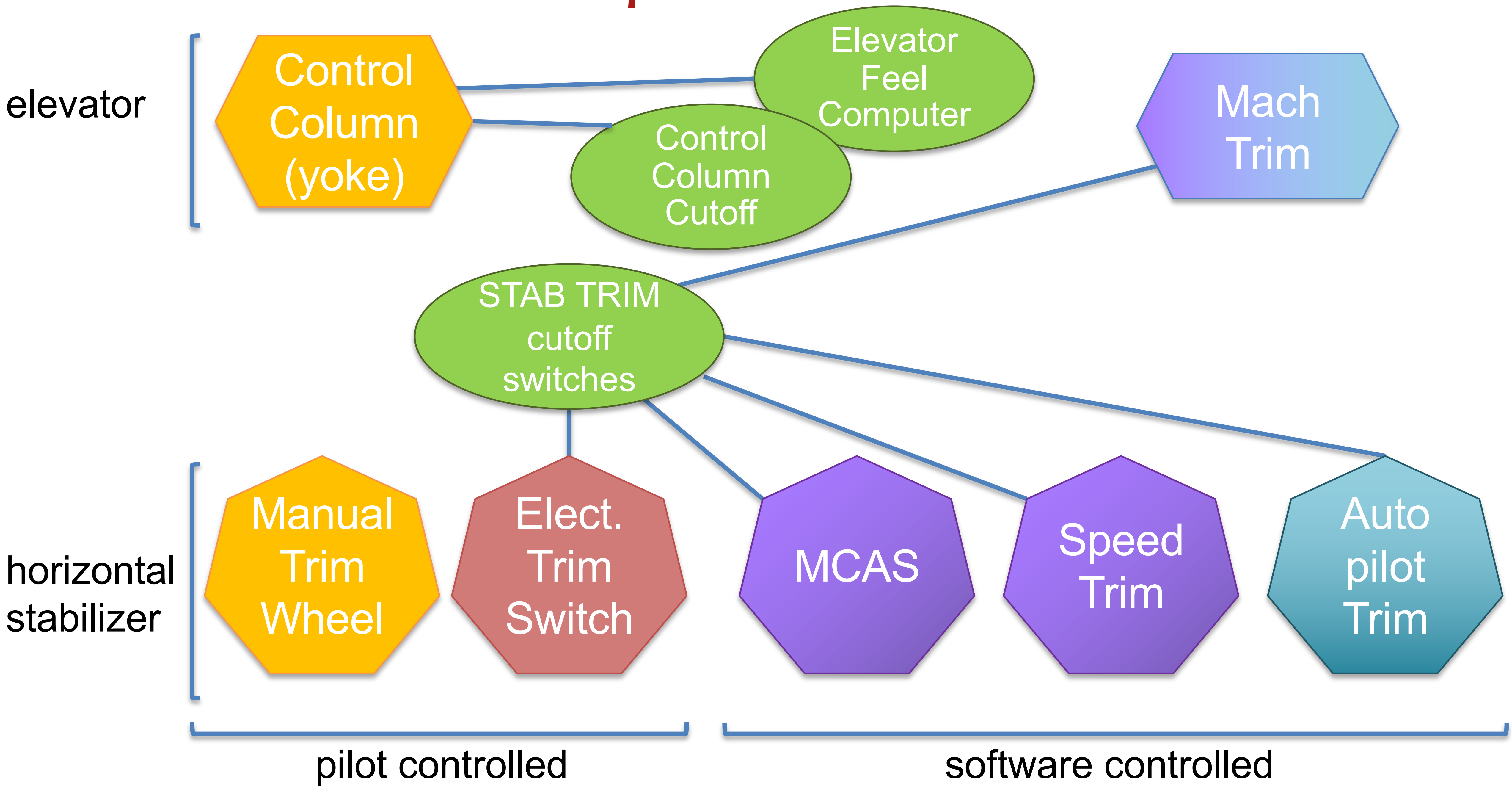
Boeing 737 MAX 8 – pitch control



Elevators
pivoting the elevators upwards creates a downward force that pushes tail down and nose up (and vice versa)

Horizontal Stabilizer
rotating the stabilizer pushes tail up and nose down (and vice versa)

features that affect pitch control surfaces



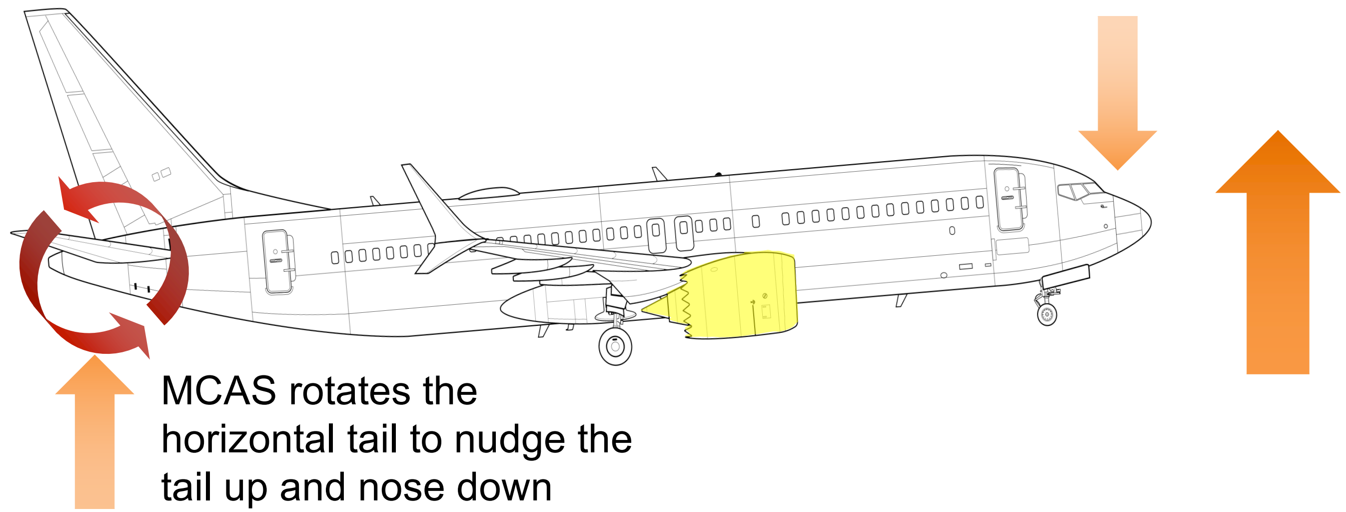
MCAS

Activates under strict conditions

- High G-force (upward acceleration)
- Angle of attack is high
- Autopilot off
- Flaps are up

Has limited impact

- Moves horizontal stabilizer at most 0.6 degrees
- Deactivates when pilot applies trim



(image derived from norebbo.com templates)

MCAS

Activates under strict conditions

- High G-force (upward acceleration)
- Angle of attack is high
- Autopilot off
- Flaps are up

Has limited impact

- Moves horizontal stabilizer at most 0.6 degrees
- Deactivates when pilot applies trim

evolves to MCAS'

Activates under looser conditions

- ~~High G-force (upward acceleration)~~
- Angle of attack is high
- Autopilot off
- Flaps are up

More powerful

- Moves horizontal stabilizer 2.4 degrees
- Deactivates when pilot applies trim

Changes to pilot cut-offs

- Cut-off switches deactivate electric trim as well as automatic trim
- Disables control column cut-off capability

MCAS is poorly communicated to pilots

feature interactions (Lion Air Flight 610)

1) Elect Trim Switch · MCAS' · Speed Trim — **can engage under the same conditions and can have conflicting actions**

2) MCAS' · Control Column (yoke) — **MCAS can trim the nose by 2.4 units per cycle, which is faster than pilot's ability to trim the nose**

- automated trim with flaps up is limited to 0.09 deg/sec
- MCAS moves at 0.27 deg/sec
- pilot's trim with flaps up is limited to 0.2 deg/sec

3) Elect Trim Switch **resets** MCAS' — **allowing MCAS to re-engage repeatedly**

4) MCAS' **inhibits** Control Column Override — **disabling the pilots' most ingrained means of stopping Automatic Trim**

feature interactions (Ethiopian Airlines Flight 320)

1) MCAS' **inhibits** Control Column Override — **disabling the pilots' most ingrained means of stopping Automatic Trim**

2) Elect Trim Switch **resets** MCAS' — **allowing MCAS to engage repeatedly**

3) MCAS' Control Column (yoke) Manual Trim — **MCAS can severely mis-trim the nose so that pilots are unable to maneuver the stabilizer appreciably nose up**

4) MCAS' Speed Trim Auto pilot Trim — **because features apply automatic trim routinely, they can mask MCAS actions**

detecting interactions (violations of feature specifications)

executable
model of
feature

property of feature

$$F_1 \models \Phi_1$$

$$F_2 \models \Phi_2$$

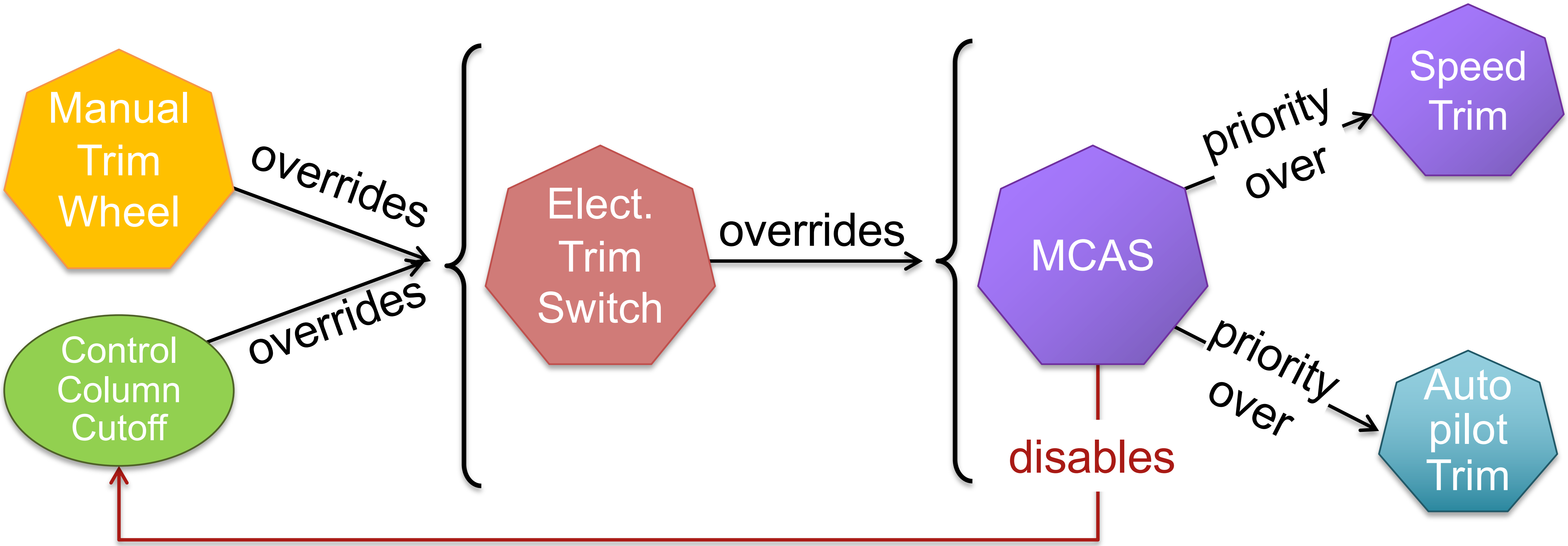
$$\vdots$$

$$F_n \models \Phi_n$$

$$\underbrace{F_1 \oplus F_2 \oplus \cdots \oplus F_n}_{\text{feature composition (= product)}} \not\models \Phi_1 \wedge \Phi_2 \wedge \cdots \wedge \Phi_n$$

detection is not always obvious

the only obvious interaction was intended



best resolution is not always obvious

› **Pat forwards all of her calls to Ana**
› **Sal calls Pat**
› **The call attempt fails (no answer)**

Whose Voicemail should activate?

- what if Pat is a sales group and Ana is a sales representative?
- what if Pat is on a long leave of absence?

nonmonotonic resolutions
(Veldhuijsen'95)

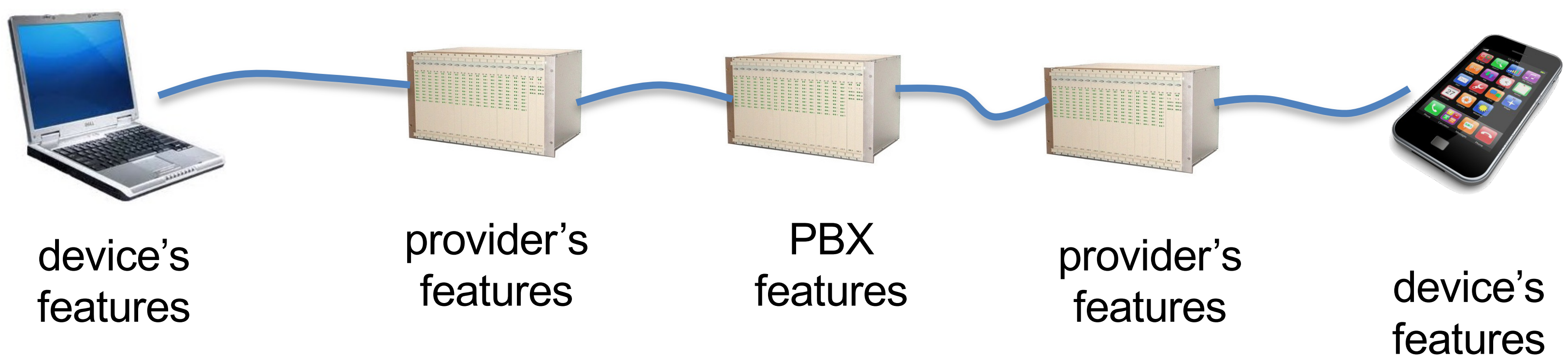
a new feature can change the requirements of existing features

- nonmonotonic extensions
  - e.g., hybrid brakes $\oplus$ anti-lock brakes
- changes to definitions of terms
  - e.g., refinement of the notion of being *busy*
  - e.g., evolution of a *call*
  - e.g., evolution of *phone directory; private numbers*
- violation of invariants / assumptions
  - *for almost any interesting invariant, there is often an interesting feature that would violate it*

the ugly: scalability

lots of features

telephony, automotive software have 1000+ features



a system of feature-rich systems

- features from multiple providers
- multiple active versions of the same feature

lots of types of interactions

control-flow
one feature affects the flow of control in another feature

data-flow
one feature affects (deletes, alters) a message destined for another feature

data modification
shared data read by one feature is modified by another feature

data conflict
two features modify the same data

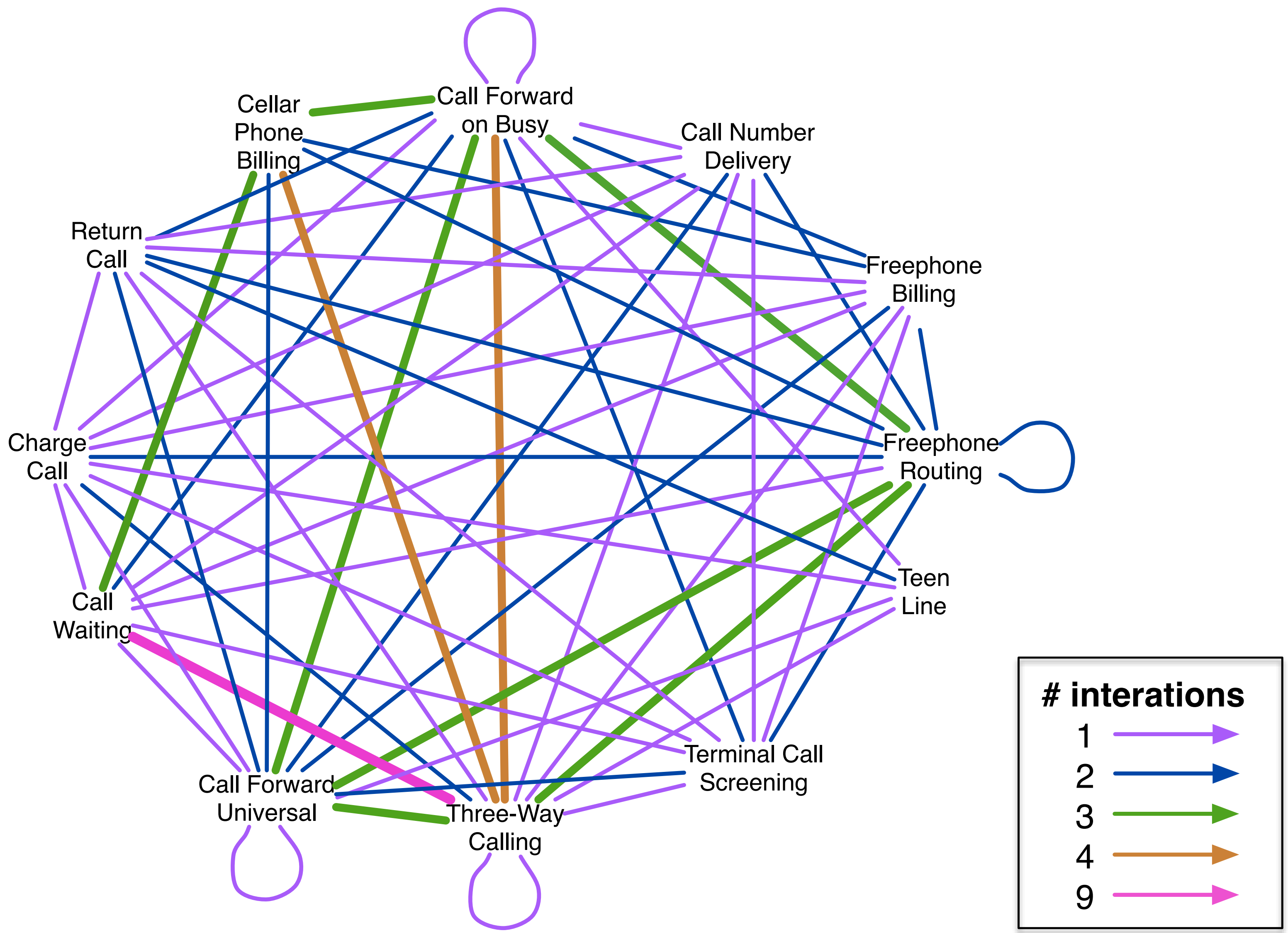
control conflicts
two features issue conflicting actions

assertion violation
one feature violates another feature's assertions or invariants

resource contention
the supply of resources is inadequate, given the set of competing features

lots of interaction instances

Griffeth, Blumenthal, Grégoire, Ohta, "A feature interaction benchmark for the first feature interaction detection contest., Computer Network, 2000.



introduced in several phases

Bowen, SETSS'89

[requirement] understanding / specifying how features ought to interact

[requirement] the number of interactions (and resolutions) to consider grows exponentially with the number of features

[design] more interactions introduced during design due to sharing of resources, I/O devices, protocol signals, etc.

[implementation] near-commonalities among features leads to questions about how to effectively reuse software components

[test] the sheer number of possible interactions and intended resolutions to be tested lengthens the testing phase

resolutions as new requirements

$$F_{\mathbf{1}} = f_1$$

$$+\ e_{f_2} + e_{f_3} + e_{f_4} + e_{f_5} + e_{f_6} + e_{f_7} + \ldots + e_{f_n}$$

$$+\ e_{f_2 f_3} + e_{f_2 f_4} + \ldots + e_{f_2 f_n} + \ldots + e_{f_{n-1} f_n}$$

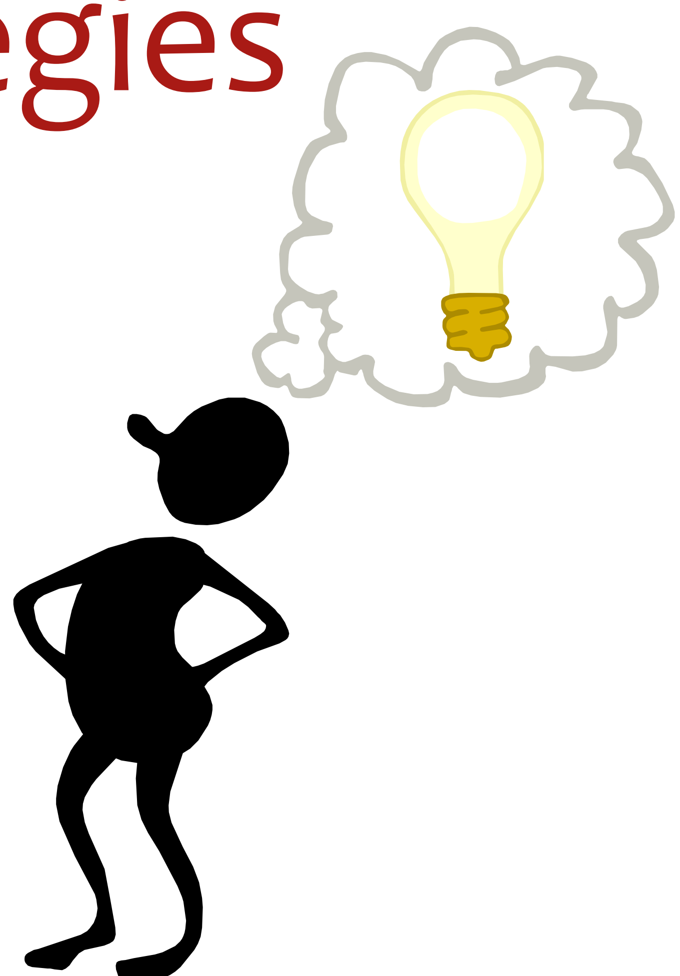
$$+\ e_{f_2 f_3 f_4} + e_{f_2 f_3 f_5} + \ldots + e_{f_{n-2} f_{n-1} f_n}$$

$$\ldots$$

$$+\ e_{f_2 f_3 f_4 f_5 f_6 \ldots f_n}$$

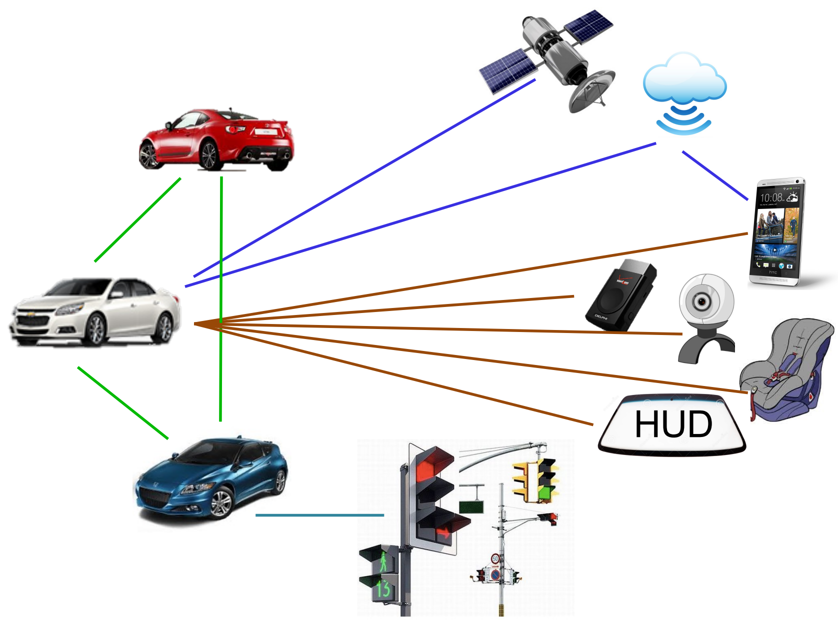
this is exactly the chore that feature-orientation was meant to avoid!

in search of general strategies



degrees of resolution perfection

- fixed set of features
- pre-determined selection of features
- static integration
- perfect coordination possible

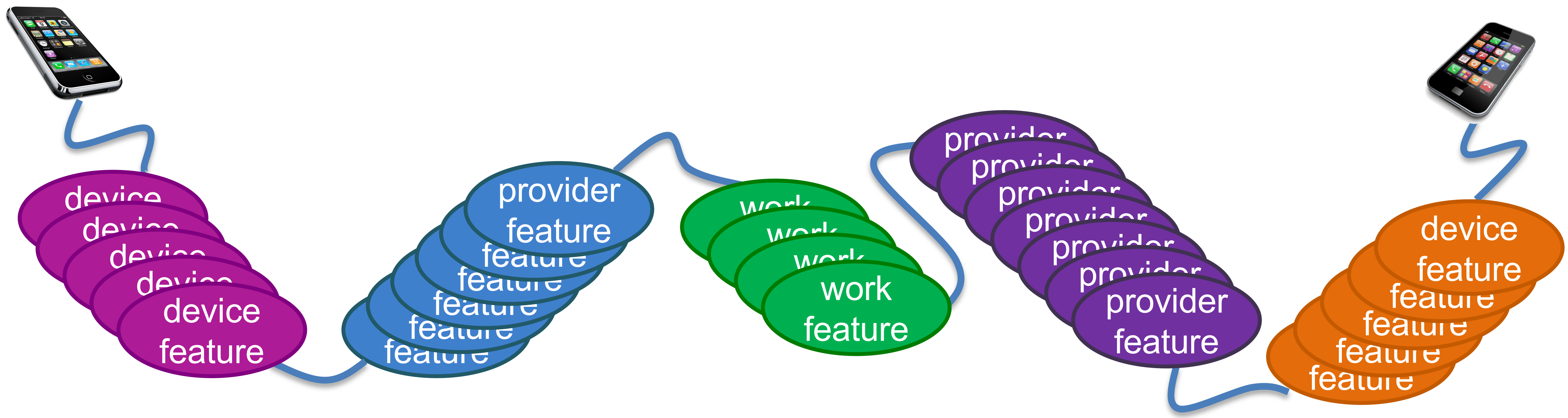


- changing set of features
- configurable
- set of static integrations, dynamic upgrades
- safe, predictable, “good enough” coordination

- unlimited features
- user-defined selection of features
- dynamic integration
- loose coordination

example #1 - serialization

Distributed Feature Composition [Jackson, Zave, TSE'98]

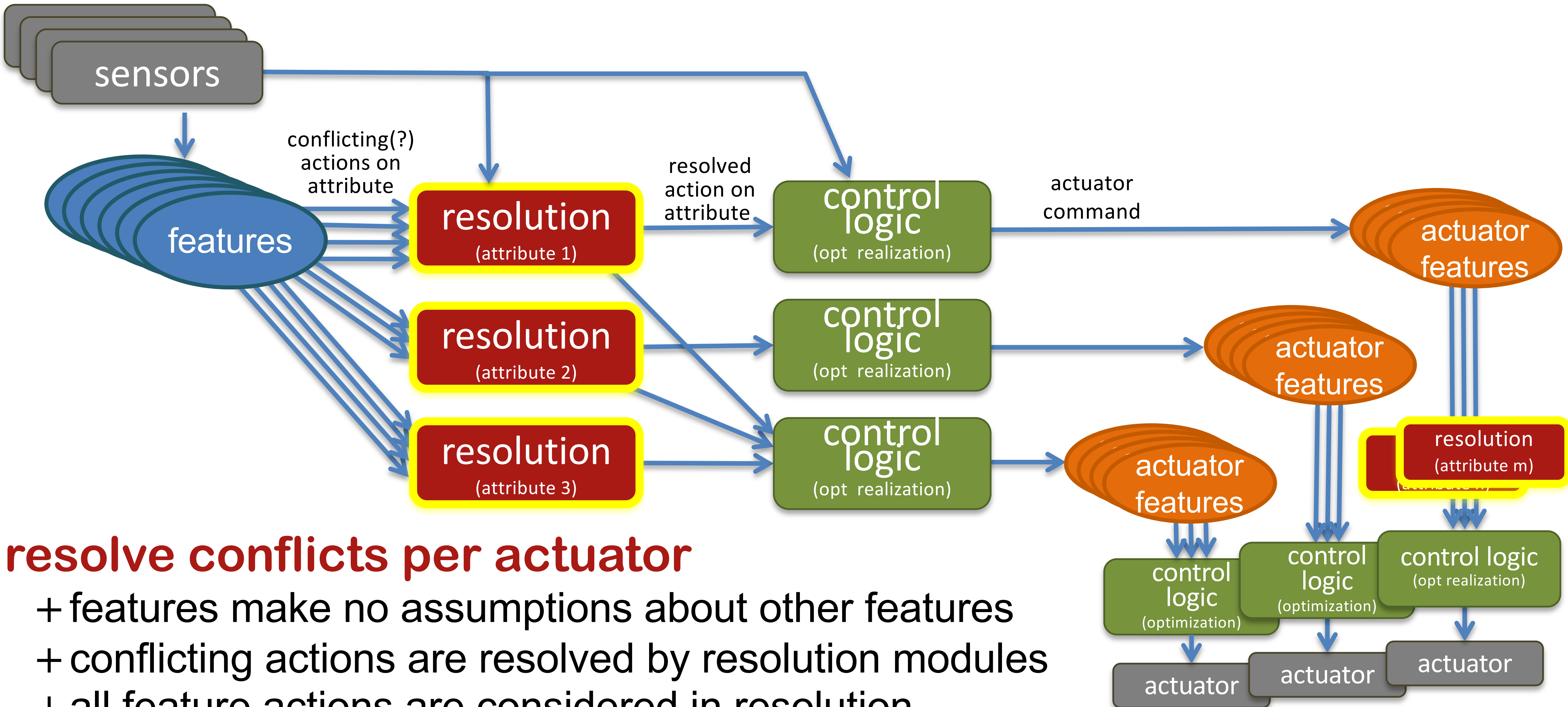


constrain information flow: features react to signals in sequence

+ features make no assumptions about other features
+ avoids simultaneous reactions to the same event
+ conflicts are resolved through serialization
+ feature ordering realizes a priority scheme

example #2 – resolution modules

Continuous Variable-Specific Resolution of Feature Interactions [Zibaeenejad, Zhang, Atlee, FSE'17]

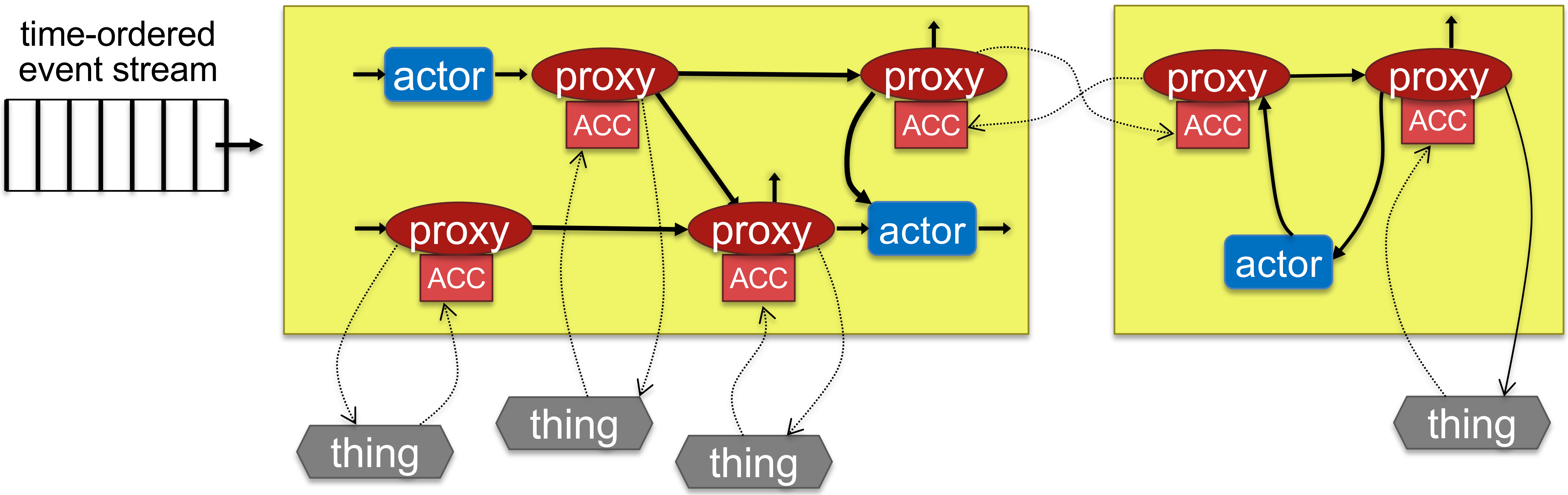


resolve conflicts per actuator

+ features make no assumptions about other features
+ conflicting actions are resolved by resolution modules
+ all feature actions are considered in resolution
+ resolution strategies are programmable (can be variable- or actuator-specific)

example #3 – device atomicity + actor coordination

A Component Architecture for the Internet of Things [Brooks, Jerad, Kim, Lee, Lohstroh, et al. Proc. of the IEEE, 2018]



Interactions with devices distinct from interactions among actors

+ devices execute concurrently, asynchronously
+ devices' events are handled atomically (asynchronous atomic callback functions)
+ actors and proxies coordinated by timed event-driven semantics
+ devices, proxies, actors make no assumptions about each other

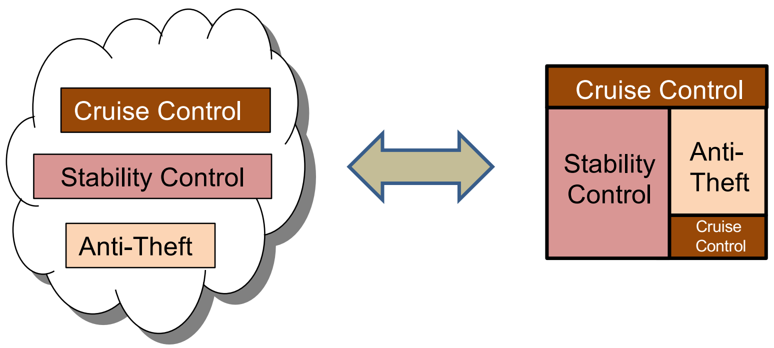
verifying coordinated features

$$\underbrace{F_1 \oplus F_2 \oplus \cdots \oplus F_n}_{\text{feature coordination}} \models ?$$

verifying that the behaviour of features coordinated by an architecture is safe, predictable, good enough

feature-oriented software development

feature: a unit of added-value



stakeholders' mental model of system

feature-oriented software system

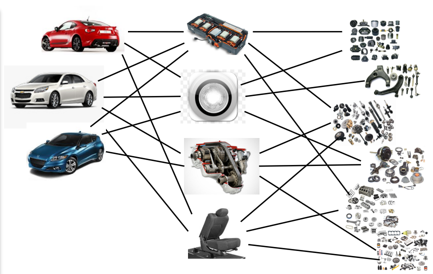
death by exceptions

$$F_1 = f_1$$
$$+ e_{f_2} + e_{f_3} + e_{f_4} + e_{f_5} + e_{f_6} + e_{f_7} + \ldots + e_{f_n}$$
$$+ e_{f_2 f_3} + e_{f_2 f_4} + \ldots + e_{f_2 f_n} + \ldots + e_{f_{n-1} f_n}$$
$$+ e_{f_2 f_3 f_4} + e_{f_2 f_3 f_5} + \ldots + e_{f_{n-2} f_{n-1} f_n}$$
$$\ldots$$
$$+ e_{f_2 f_3 f_4 f_5 f_6 \ldots f_n}$$

this is exactly the chore that feature-orientation was meant to avoid!

feature coordination

- fixed set of features
- pre-determined selection of features
- static integration
- perfect coordination possible



- changing set of features
- configurable
- set of static integrations, dynamic upgrades
- safe, predictable, "good enough" coordination

- unlimited features
- user-defined selection of features
- dynamic integration
- loose coordination

verifying coordinated features

$$\underbrace{F_1 \oplus F_2 \oplus \cdots \oplus F_n}_{\text{feature composition}} \models ?$$

verifying that the behaviour of features coordinated by an architecture is safe, predictable, good enough