

# TECHNOLOGY FOR A MOVING WORLD

## ESI Symposium – Integrating Systems

Kevin de Jong

27<sup>th</sup> of September 2022

**TomTom** 

# Kevin de Jong

*Engineering Manager*

For the last **14 years**, I have been navigating the **Automotive landscape** with **TomTom**, from software engineer to delivery manager, and finally, managing a team tasked to **improve the TomTom Developer Experience**.



**TomTom is The  
leading  
independent  
location  
technology  
specialist**



Highly accurate maps



Navigation software



Real-time traffic and services

# History of Automotive

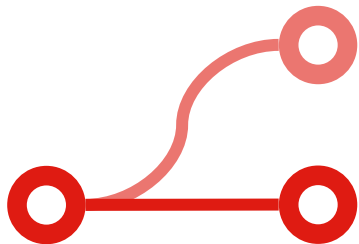
The era of Sat Nav



1994+

# History of Automotive

## The beginning of Automotive



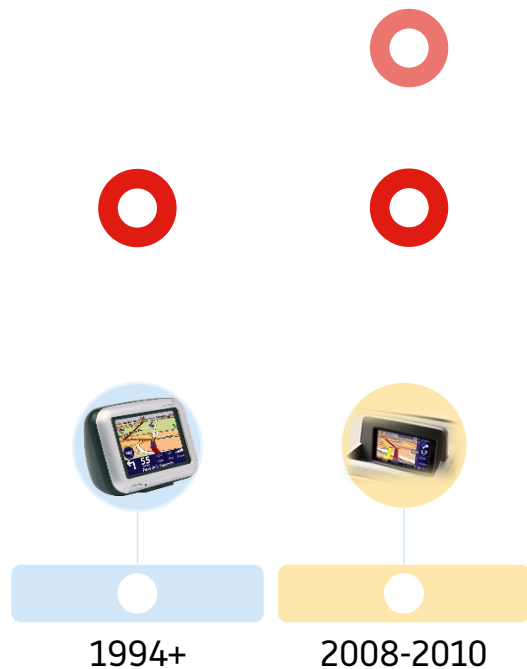
1994+



2008-2010

# History of Automotive

Consumer vs automotive quality



# History of Automotive

The rise of connectivity



1994+

2008-2010

2010-2012

# History of Automotive

Catch-up with latest innovations in Consumer market



1994+



2008-2010

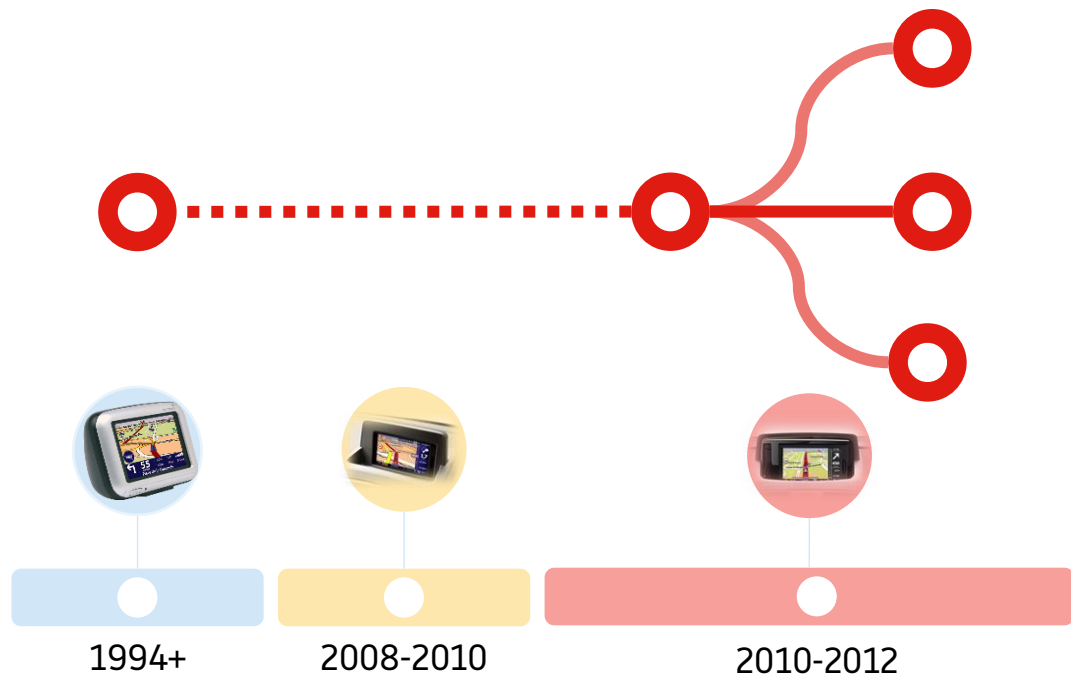


2010-2012



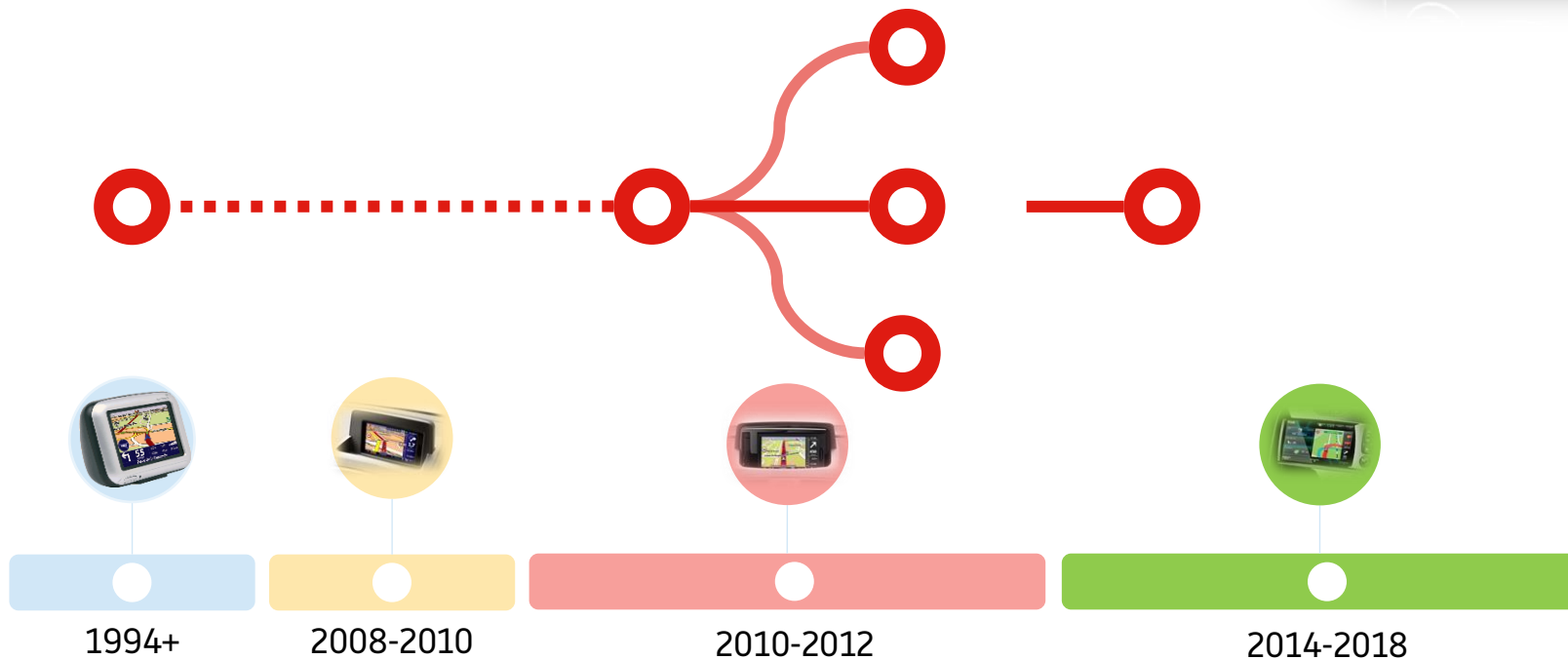
# History of Automotive

Automotive products derived from an “Automotive”-navigation engine



# History of Automotive

## In-vehicle Infotainment on Android



# History of Automotive

Again, catch-up with latest innovations in Consumer



1994+



2008-2010



2010-2012



2014-2018

# History of Automotive

Turning-point within Automotive



1994+



2008-2010



2010-2012

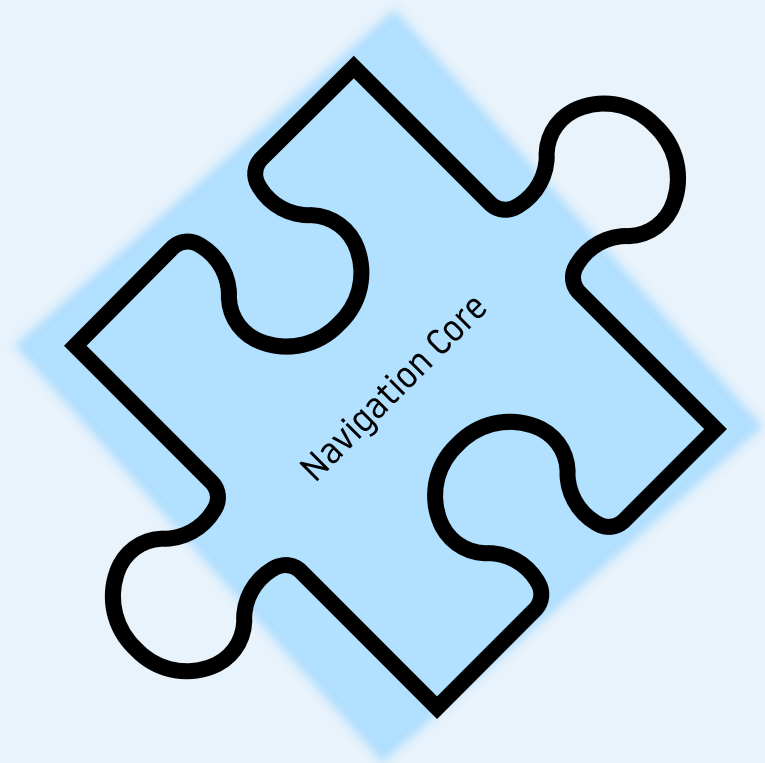


2014-2018

# Navigation Core

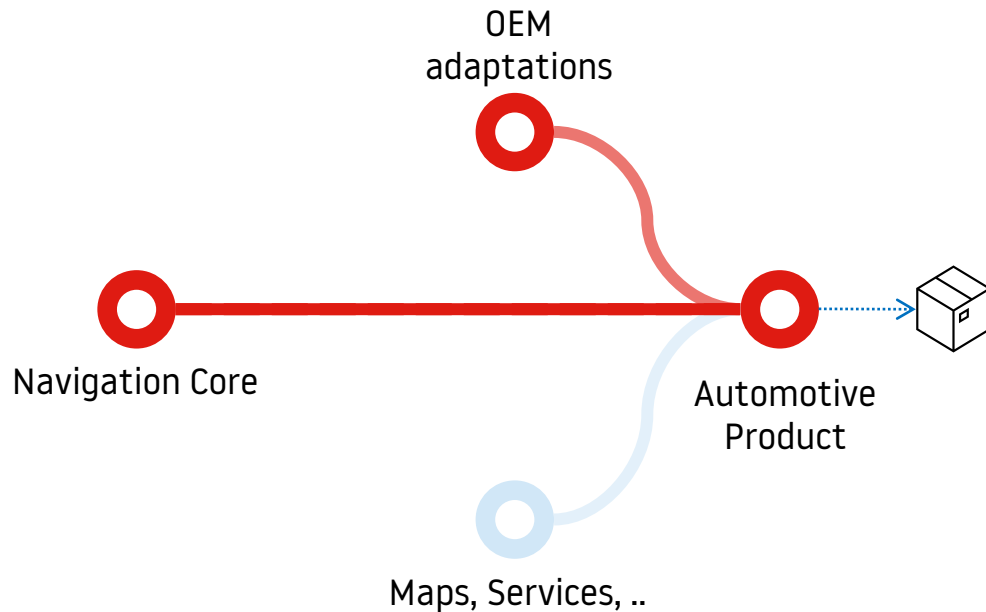
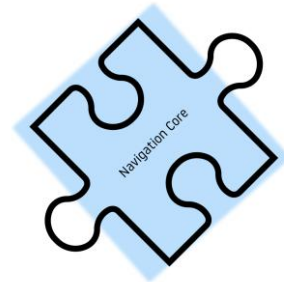
*One core to rule them all*

*During our rise in Automotive, we have been dependent on our 'Navigation Core' - a single solution providing a complete **end-to-end navigation application**.*



# Delivery Chain

Simple integration strategy



**Delivery Frequency:** once upon start of production, followed by once per 5 years

**Mean-Time-To-Restore:** between 6 and 12 months

# Key takeaways (till '14)

## Based upon the early rise in Automotive

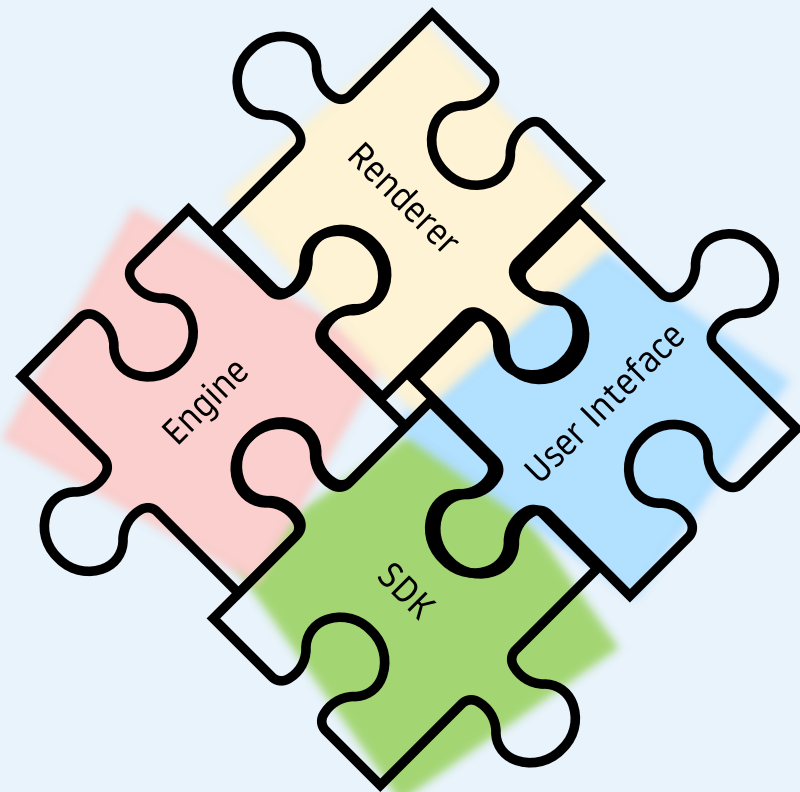
- Our **consumer products** have been the source of **rapid innovation**
  - Next-generation Automotive products relied on these innovations, but **at the cost of hardening again**.
  - Automotive has been lagging due to **strict requirements and regulations**.
- The **impact of a failure** is significantly higher in Automotive.
- **Agile- vs. V-model** methodology
  - Software development was using Agile.
  - Project management applied Agile within V-Model project execution.
- There has been **limited ability to upgrade software** in our Automotive products.
  - Even though we introduced **in-vehicle software updates** using consumer methods (i.e. SD Card) and TomTom HOME desktop application - often a **recall action** would be required to upgrade software
  - The whole value stream would require recertification and validation (incl. in-vehicle validation by OEMs), **requiring months**.



# Navigation Kit

*Extended flexibility*

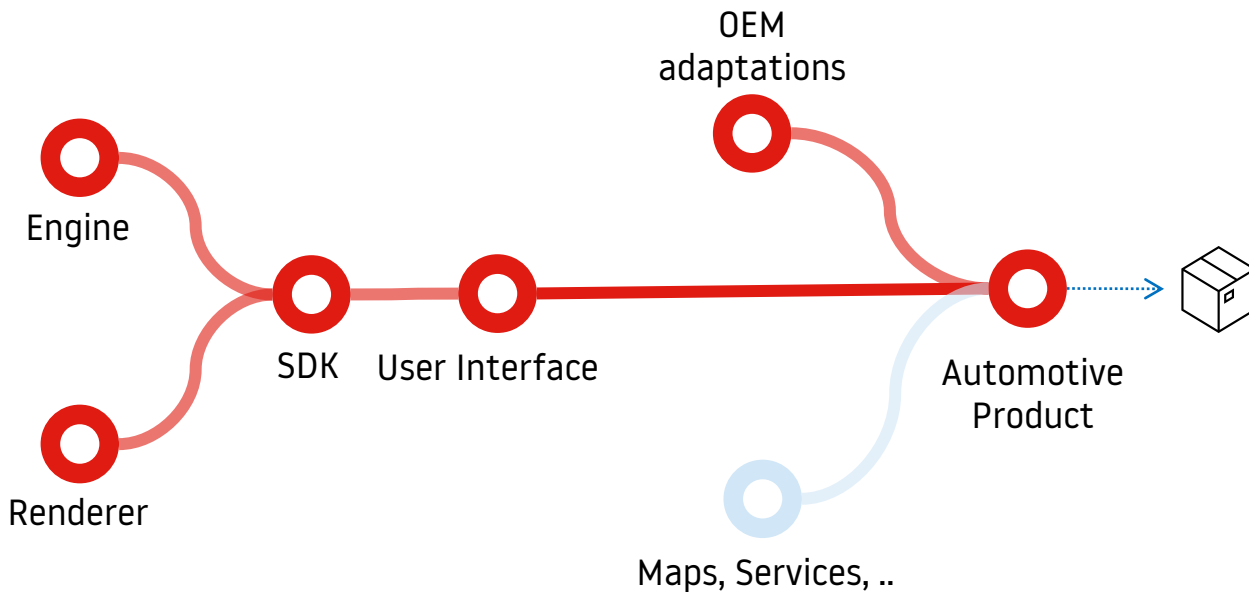
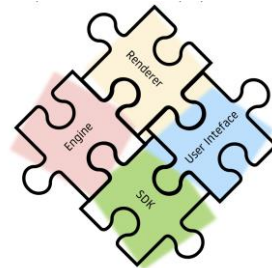
Extended our product offering by dividing our Navigation Core in **multiple “off-the-shelf” products**





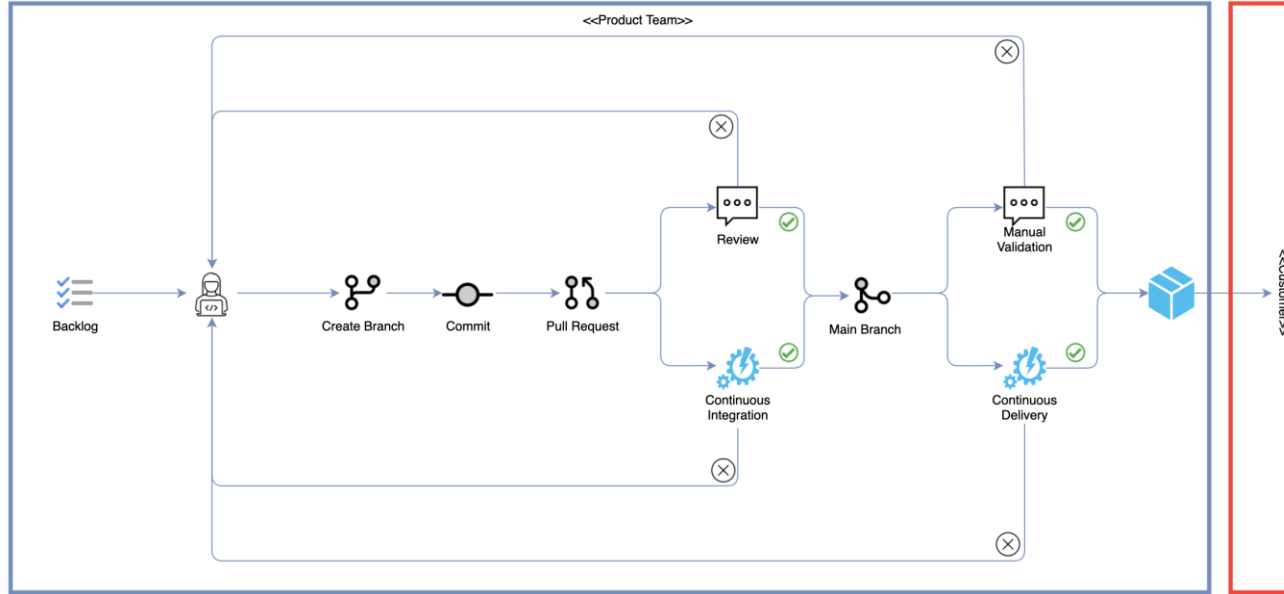
# Delivery Chain

NavKit1



# Continuous Integration

*Engineering practice aimed at "merging" code changes into the main branch as quickly as possible*



## Continuous Integration

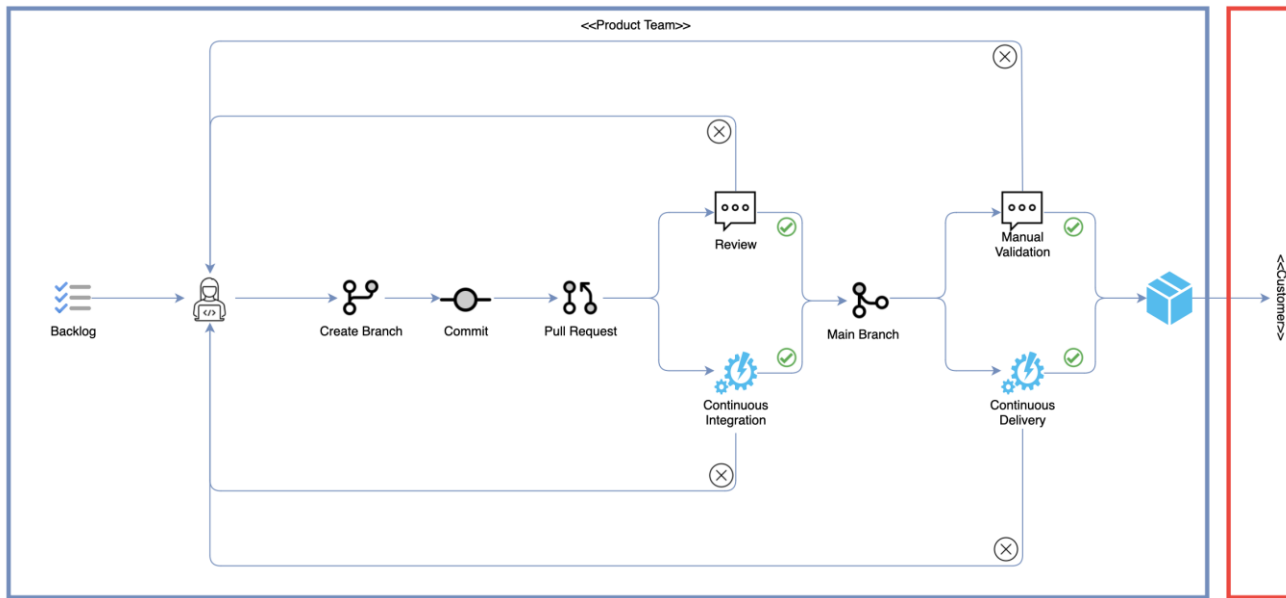
**Component scope** - Code Review, Linters, Unit Testing

**Cycle Time:** +/- 60min

**Deliverable:** Source Code and/or Engineering package (only to be consumed by the respective development team)

# Continuous Delivery

*Automatically deliver code changes to a (pre-)production environment*



**System scope** - Functional Testing, Performance Testing, (Full) Static Code Analysis, (Full) Static Application Security Testing

**Cycle Time:** +/- 1h30min

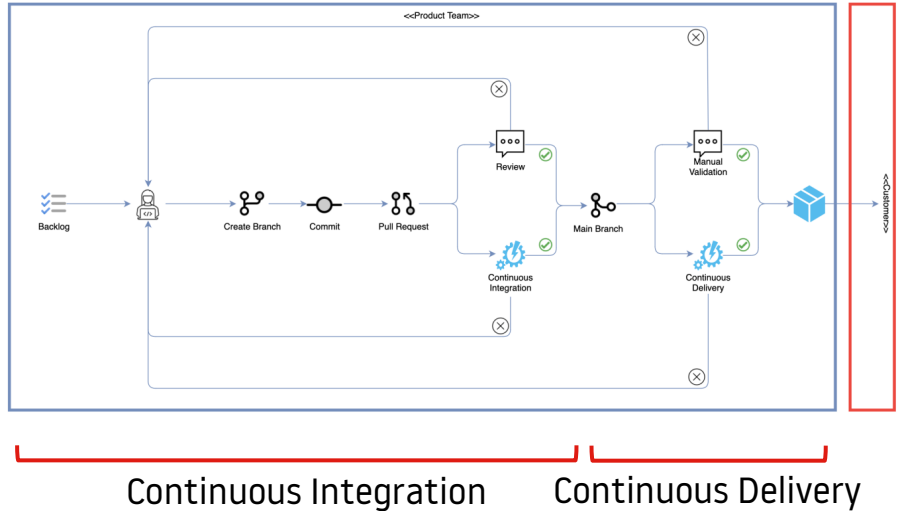
**Deliverable:** "Potential shippable product"

Continuous Delivery

# Continuous Delivery

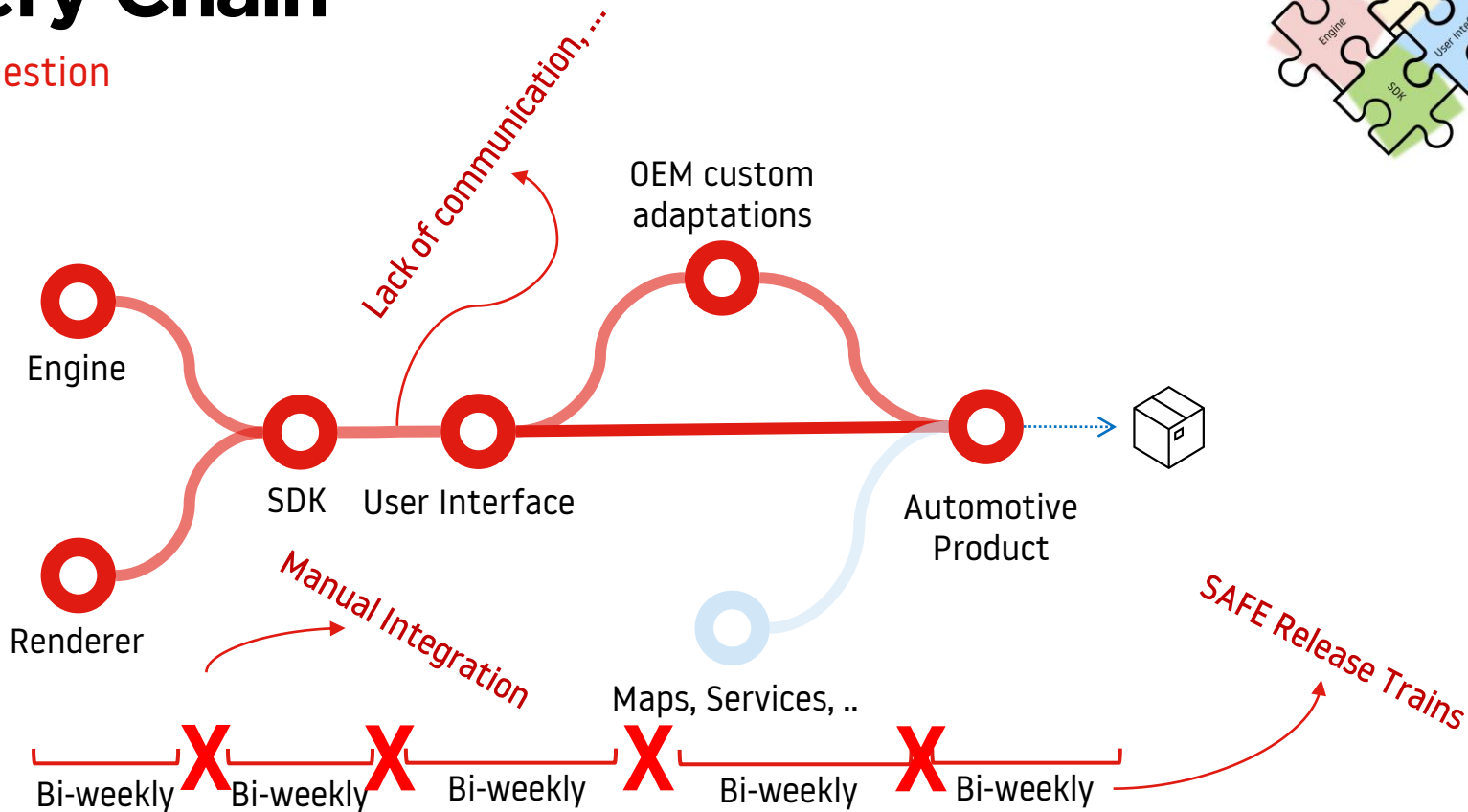
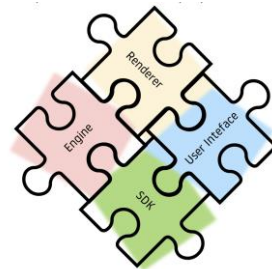
## Following best-practices

- **Fail often and fail fast** - ensure that changes are integrated often while failures are addressed before merging modifications to your main branch
- **Keep the build green at all times** - failures can still happen on your main branch (i.e., flaky testing, extensive duration tests not executed as part of a Pull Request, ..).
- **Feedback loops** - Adding notifications for failures is crucial to maintain a green environment.
- **Create new tests as part of development** - as test automation is the pillar of Continuous Integration, it is equally essential to ensure that the creation of automated tests is part of the standard development flow



# Delivery Chain

Release congestion



# Continuous Delivery (cont)

## Continuous investment

- Continuous Integration:

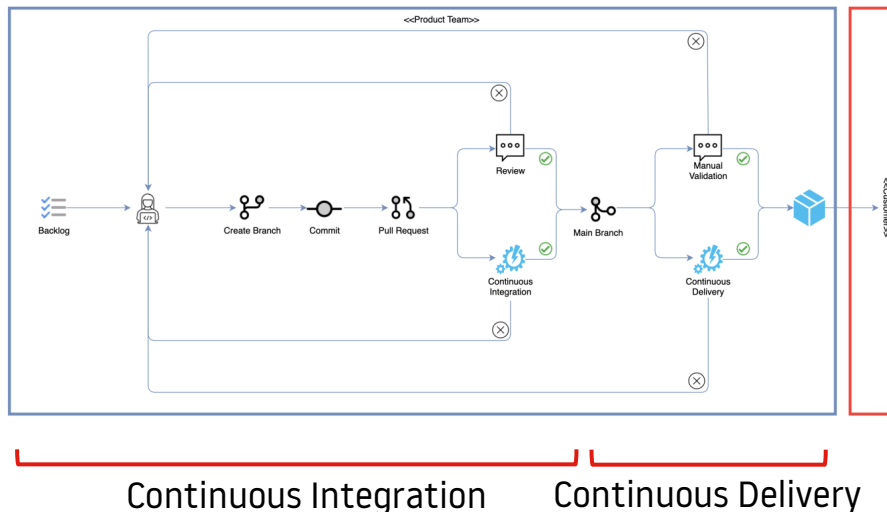
- Continuous (automated) **intake strategy**
- Reduce compilation time (*incremental builds, splitting components, optimizing build system, caching, ...*)
- Introduce feedback loops from SCA/SAST tools for early detection

- Continuous Delivery:

- **Contract Testing**
- **Contract Testing**
- **Contract Testing**

- And more...

- API Management
- Package management
- Metrics, Monitoring, Alerting, ...



# Stakes are increased

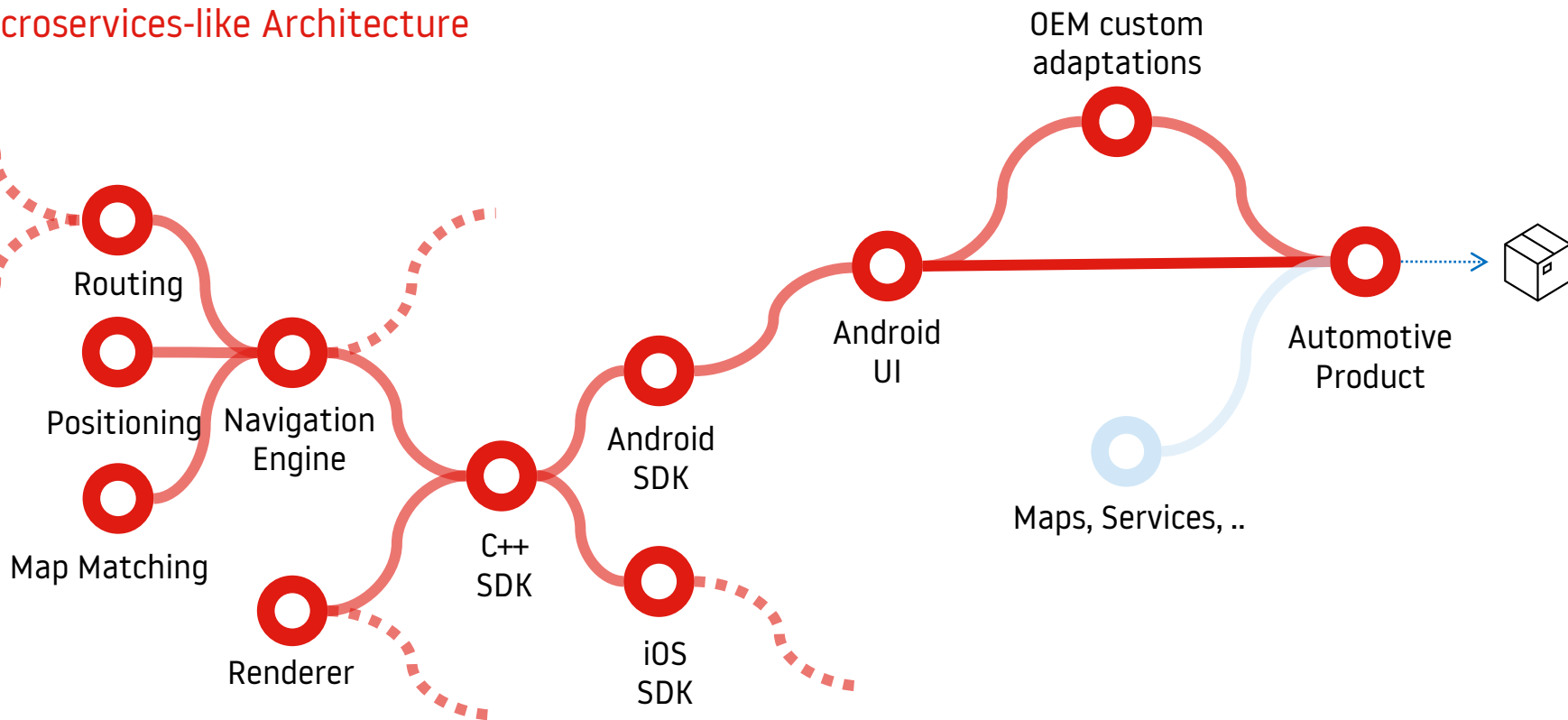
Customers are expecting first-in-class user experiences,  
The Automotive industry is starting to accept Software as a Service,  
Online only- and Hybrid- solutions are in high demand,  
while Enterprise business is rapidly expanding.



*Please help make the  
mythical man-month a  
reality*

# Delivery Chain

## Microservices-like Architecture





**“Organizations, who design systems, are constrained to produce designs which are copies of the communication structures of these organizations.”**

---

Conway's Law

# Product Teams

*It's not all about tools and technology, but also about **people**.*



# Product teams

Organizational change based upon Marty Cagan's


*“Empowered – Ordinary people, Extraordinary products”*

- **Serve the business** – teams should be accountable for solving **customer problems** (in alignment with the company vision) instead of implementing a feature backlog.
- **Holistic solutions** - ensure solutions are **valuable, viable, usable, and feasible** instead of focussing purely on software design and code.
- **Product as a Business** - Product Managers manage their product as if they are **the CEO of their own company**, instead of managing project execution
- **Outcome focussed** - Hold the team **accountable** for their results. Measure **outcome**, not output nor roadmaps

Team Topologies



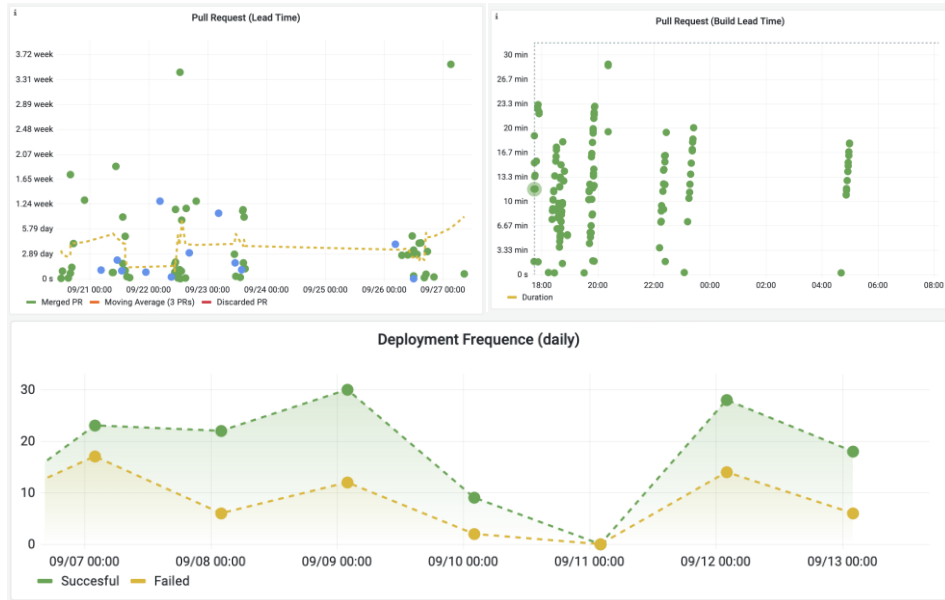
Social Technical Congruence??



# Continuous Delivery vs Product Teams

or: how to never forget the word “continuous”

- Design the Delivery Chain around your **value stream**
- Agreements need to be made on the **hand-over** between products by the value stream.
  - Eliminate the need for product-specific requirements from your daily workflow
  - Lead time should be determined by the time required to build your product, not the E2E delivery chain.
  - Failure in a downstream dependency should not block your development flow
  - Rely on Intake strategies for managing upstream dependencies
- Boundaries define **accountability** for your product team(s)
  - You are accountable for the Quality constraints of your product
  - Your (internal) customer needs to address failed intakes of your product
- The **local development** environment needs to be prioritized
- Employ **Dev(Sec)Ops** practices within the Product Team
- Define **metrics of success** and act upon them



# Continuous Deployment

---

The current evolution in Automotive