VOLVO

Why are we so bad at software development?

And how can we fix it?

What are we talking about today

Culture

Our values, beliefs, and behaviors that define how we interact with each other

Strategy

Our organization, processes, plans, and tools we use to complete a project

Execution

Our daily project management with scheduling, resource planning, tracking, and reporting

Outcome

The project result with deliverables, quality, and time & budget outcome

Three engineering domains shaped by the process

SP SPECIFICATIONS

Documentation and artifacts defining what and how software should be built

Include requirements, specifications, and tests

FE FEATURES

Functionality providing customer value as a part of a larger solution

Feeds into a larger solution

PR PRODUCT

Software developed with multiple planned releases

Life cycle managed separately from carlines

Culture that sets the foundation

SP

Isolated specification management

- Requirements and system specifications are created in isolation from dev teams
- Seen as static foundation of all development work

FE

Siloed Features

- Feature specified, planned, and tracked in isolation
- Lack of cross-feature integration & harmonization -> Conway’s law

PR

Carlines’ needs dominates

- Long-term reuse is sacrificed to meet start of production
- Product mindset seen as risk for start of production

Strategies that shape project environments

SP

Lack of design refinement process

- ASPICE SWE.1-6 iterated over only once
- Leads to immature Big Design Up Front (BDUF)

FE

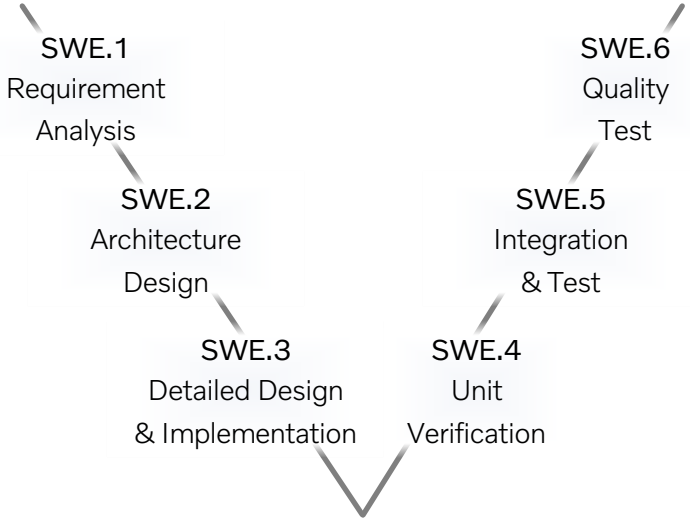
Lack of feature prioritization

- Overloaded dev teams trying to deliver to multiple function owners

PR

No software product KPIs

- No incentive for ROI after start of production
- Software org gets treated as scale-out partner



Chaotic projects

SP

No definition of done

- Lack of atomic, testable requirements
- No traceability between artifacts

FE

Panic descoping

- Feature are dropped when they are almost done.

PR

No planning beyond Start of Production

- Focus on feature completion
- No protected long-term development

We have all been here

Late feature completion leads to...

late integration

late-stage design issues

project panic

developer burnout

Conclusion

	Culture	Strategy	Execution	Outcome
Specifications	• Static specs	• Immature BDUF	• No definition of done	• Delays • Delivery risk • Reactive mgmt • Burnout
Features	• Siloed features	• No prioritization	• Panic descoping	
Product	• Carline centric	• No product incentives	• SOP obsession	

VOLVO

Core Engineering Principles

Be sympathetic to the needs of others

- Mechatronics have different requirements than software

Trust your colleagues

- There are several different paths to your common goals

Balance the now with the future

- Start of production is important, but so is the time after

Change the culture to embrace core principles

SP

Assimilate agile iterations into engineering mindset

- Embrace that specification evolves with implementation
- Add feature maturity metrics to dashboard and progress reports

FE

Have solutions drive features

- See features as a value-adding part of a greater solution
- The solution objectives & requirements drives feature prioritization

PR

Decouple from carlines

- View software product as pre-fabricated deliverables to be integrated into the car

Express culture through changed strategies

SP

Implement incremental spec evolution process

- Iterate often across all ASPICE SWE steps

FE

Fewer stable features > More unstable features

- Specify a solution MVP that can ship once stable

PR

Decouple from carlines

- Product team delivers releases to carline integration team
- Separate software product funding from that of integration projects

Leverage strategy to execute distinct, measurable projects

SP

Go API First

- Separate API versioning from that of implementation
- API, Implementation, build, test – Manage separately

FE

Have a machine-executable definition of done

- Start with feature mockup delivered by architecture team
- Tie maturity metrics to API-, code-, and test velocity

PR

Decouple from carlines

- Product delivers releases to integration teams, who delivers to carlines
- Product has support inside integration team

We can all be here

MVP delivered on time leads to...

left-shifted integration

left-shifted design issues

space for tech debt management

efficient development culture

COVESA's role

Enable best software culture, strategies, and execution

- Provide an anti trust-proof forum for collaboration
- Create tooling that enables API-First principles
- Standardize on/off-board vehicle services
- Facilitate reference vehicle service implementations

VOLVO

Thank you