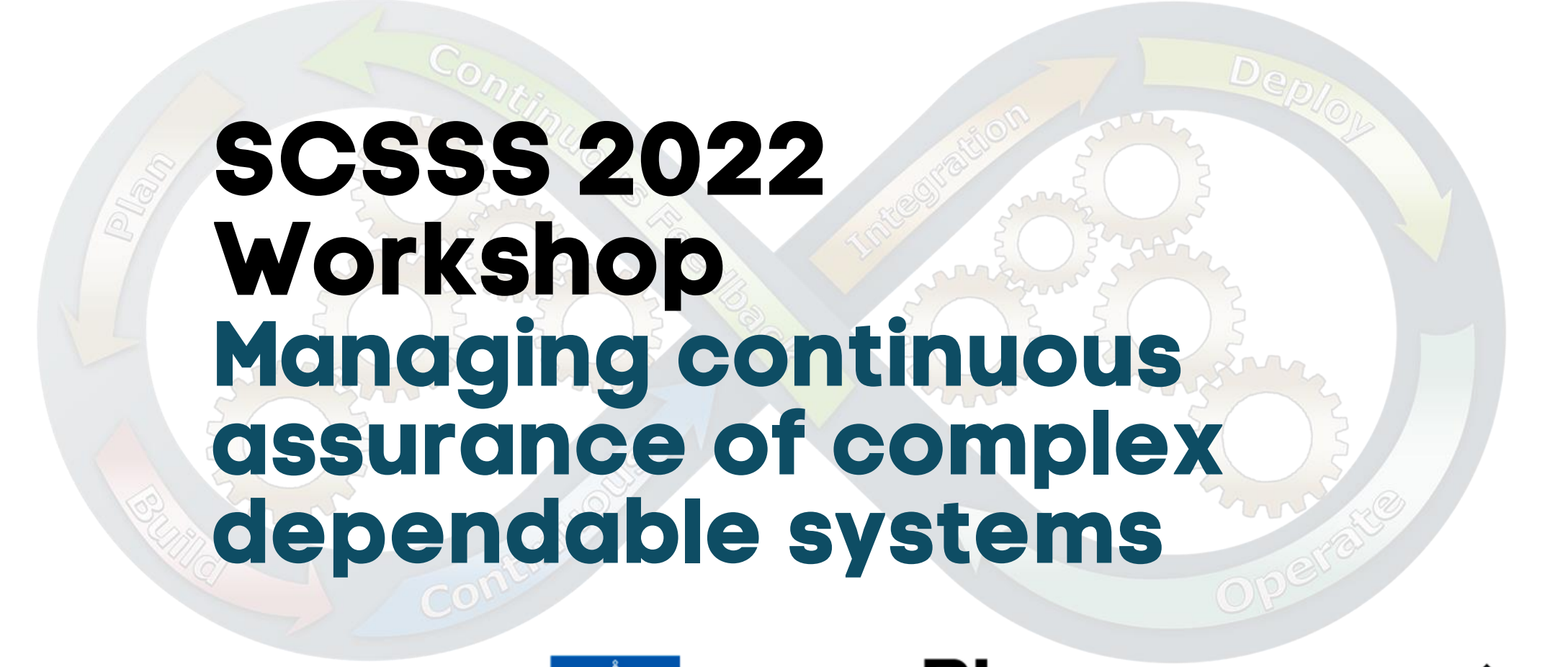
SALIENCE4CAV

Safety Lifecycle Enabling Continuous Deployment for Connected Automated Vehicles

SCSSS 2022 Workshop

Managing continuous assurance of complex dependable systems



AGREAT COMENTOR Epiroc KTH VETENSKAP OCH KONST qamcom RI. SE semcon veoneer zenseact

Motivation

Systems where safety and cybersecurity assurance is vital are increasing in complexity amid a growing business demand for faster update cycles. *(than traditional development: years)*

- Adaption to changing customer demands and business needs
- Gradual improvement of functions
- Adaption to changes in operational conditions
- Security fixes / protection against new threats

- *Months/year*
- *Months*
- *Weeks*
- *Days (hours?)*

Challenges

- Ability to manage frequent updates in a complex product
- Managing collection of field data for feedback
- “New” technology, e.g., ML components, collaborative functions
- **… and still with safety/security assurance in sync with development**

Topic for today:
Managing continuous assurance of complex dependable systems

SALIENCE4CAV
Safety Lifecycle Enabling Continuous Deployment for Connected Automated Vehicles

RI.SE

Agile Development

- Deliver the right product
  - Incremental development
  - Customer feedback
  - Frequent delivery
- Used since ~2000
- Increasing use in domains with dependability requirements
  - **Is it implemented in an efficient and effective manner given assurance needs?**

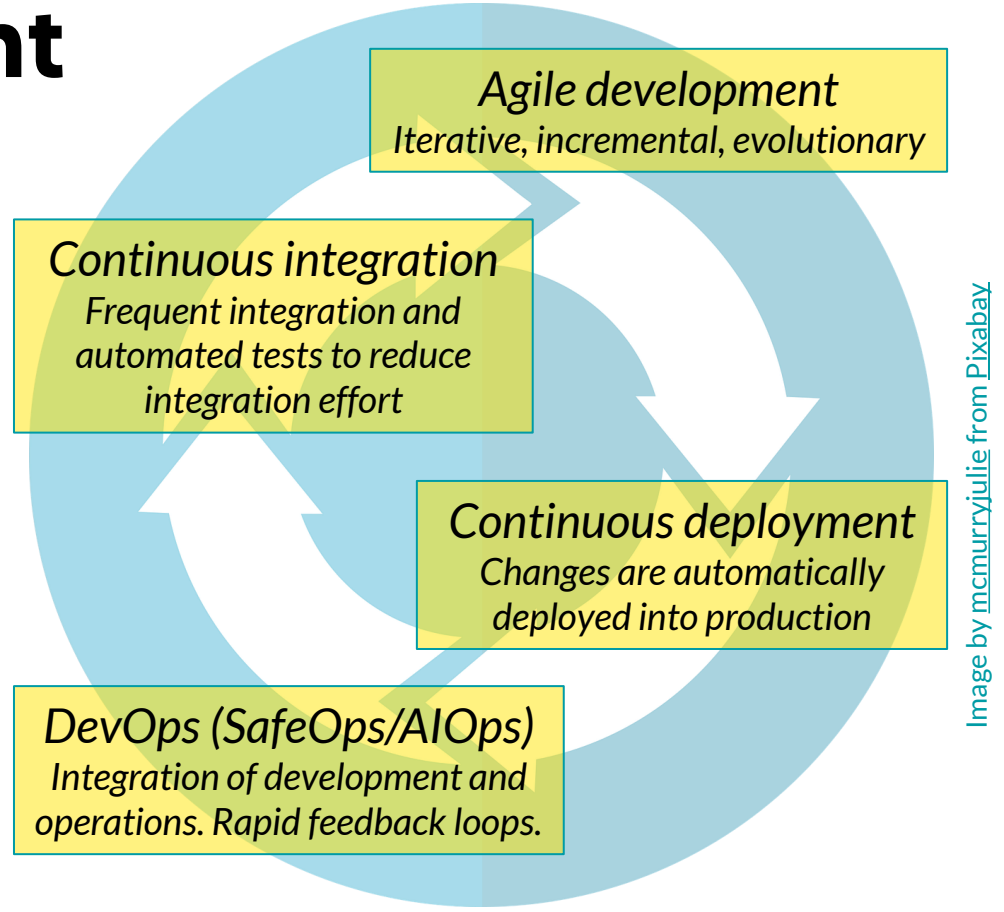


Image by mcmurryjulie from Pixabay

Where do we stand today?

- This is not a new topic…
  - SafeScrum (2012)
  - SafeOps (2020)
  - Continuous assurance cases (2019)
  - …

- Component-based
- Safety contracts
- Modular assurance cases
- Enable automation
- Continuous assessment

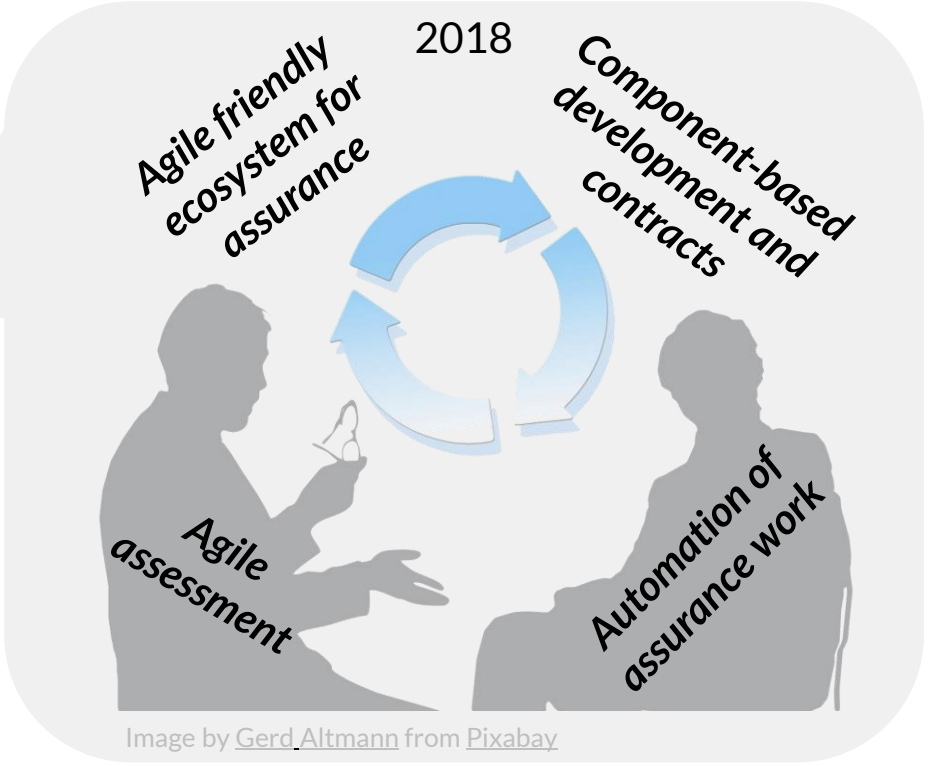


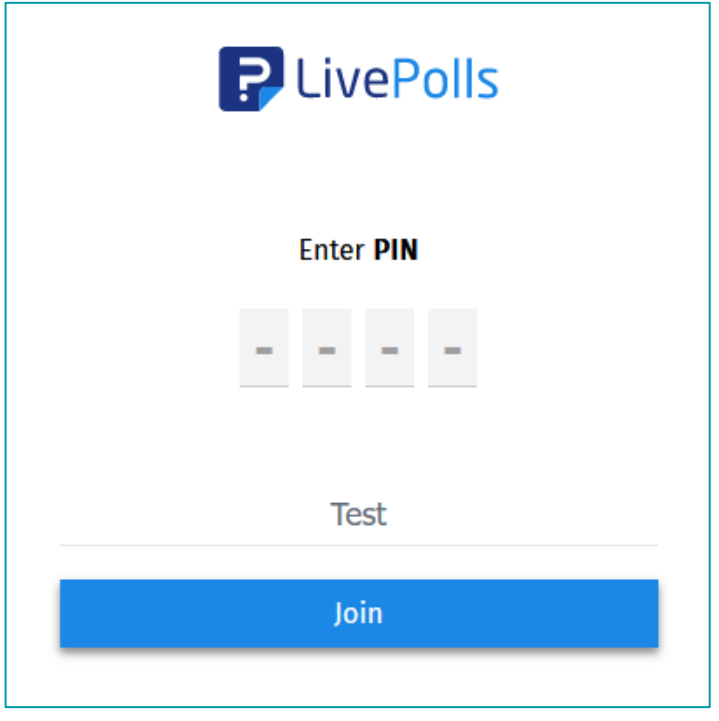
Image by Gerd Altmann from Pixabay

SALIENCE4CAV
Safety Lifecycle Enabling Continuous Deployment for Connected Automated Vehicles

RI. SE

https://www.questionpro.io/

1. (Background) Which is your main domain?
2. Do you practice continuous integration
3. Do you practice continuous deployment
4. Do you practice DevOps
5. Do you believe that adopting CI/CD can improve the development of system safety over traditional development models (less frequent releases)?
6. Do you think applying CI/CD will change safety assurance efforts?



Questions

Agenda

- **[9.30] Welcome, agenda, and introduction**
- **Challenges with CI/CD for safety-critical systems**
  - World café #1 – Challenges [25 min]
  - Key insights from world café #1
- *[~ 11.00] Coffee and sandwiches in the networking area [15 min]*
- **Potential solutions**
  - World café #2 – Solutions [25 min]
  - Key insights from world café #2
- **Wrap-up**
- *[12.30] Lunch @ L's Resto*

Discussion format

- World café style discussions
  - Form groups around tables
  - Questions available at tables
  - Select someone to take notes!
- We will collect notes
- Summary report made after workshop
  - Will be posted at: http://salience4cav.se/

Small group discussion for knowledge-sharing around pre-defined questions. Insights shared in large group.

Leave us your email if you want notification when report is ready.

References

- Warg, F., Blom, H., Borg, J., & Johansson, R. (2019, October). Continuous deployment for dependable systems with continuous assurance cases. In *2019 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)* (pp. 318-325). IEEE.
- Stålhane, T., Myklebust, T., & Hanssen, G. K. (2012, June). The application of Safe Scrum to IEC 61508 certifiable software. In *11th International Probabilistic Safety Assessment and Management Conference and the Annual European Safety and Reliability Conference* (Vol. 8, pp. 6052-6061).
- Fayollas, C., Bonnin, H., & Flebus, O. (2020, September). SafeOps: a concept of continuous safety. In *2020 16th European Dependable Computing Conference (EDCC)* (pp. 65-68). IEEE.
- Johansson, R., & Koopman, P. (2022, September). Continuous Learning Approach to Safety Engineering. In *CARS-Critical Automotive applications: Robustness & Safety*.
- Gyllenhammar, Magnus; Rodrigues de Campos, Gabriel; Törngren, Martin (2022): Holistic Perspectives on Safety of Automated Driving Systems - Methods for Provision of Evidence. TechRxiv. Preprint. https://doi.org/10.36227/techrxiv.20331243.v1

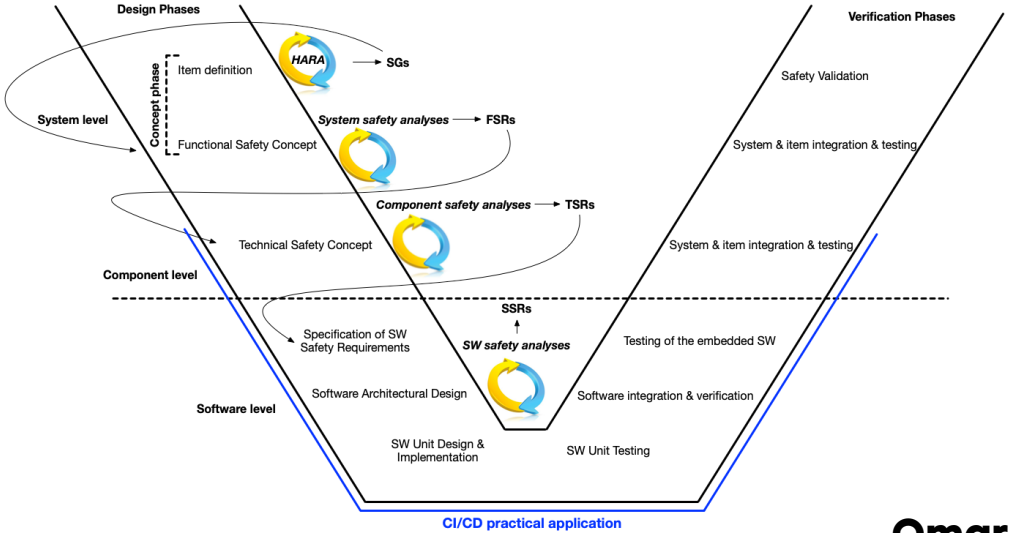
Continuous Assurance Challenges

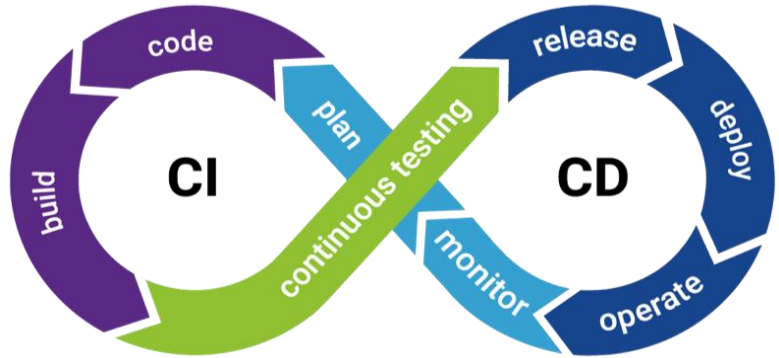
SALIENCE4CAV

Safety Lifecycle Enabling Continuous Deployment for Connected Automated Vehicles

RI.SE

Continuous- Integration & Deployment for Automotive Safety Systems





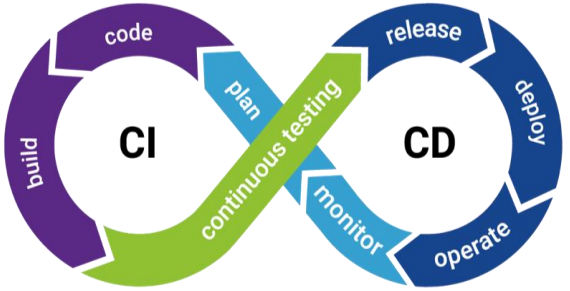
Omar Jaradat

SCSSS'22 - workshop
Nov 23, 2022
Lindholmen

SALIENCE4CAV
Safety Lifecycle Enabling Continuous Deployment for Connected Automated Vehicles

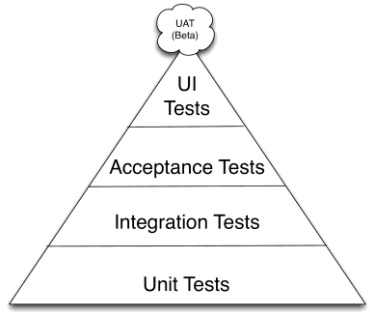
RI. SE

CI/CD



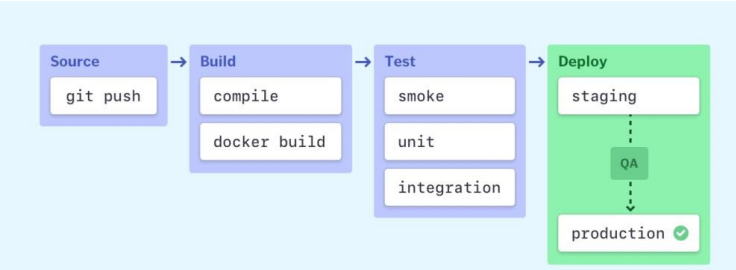
- **Continuous integration (CI)**- short-lived branches that are merged into a shared trunk several times a day where a series of automated tests give feedback about the changes introduced
  - Examples of branching strategies:
    - GitFlow
    - GitHub Flow
    - GitLab Flow
    - Trunk-based development (Our work assumes TBD as the used strategy)
- **Continuous delivery (CD)**- after continuous integration, continuous delivery prepares the software for delivery and ensures that the software can be reliably released at any time.
- **Continuous deployment**- after CI and CD, changes are automatically deployed into production by a fully automated process.

Continuous Testing



- **Unit tests**- to verify single parts of the application. This isolated part of the codebase is referred to as a unit.
- **Integration tests**- unit tests focus on an individual unit and thus may be insufficient by themselves, integration tests ensure that multiple components work together correctly and test how parts of the application work together as a whole.
- **Functional tests**- these tests make sure that the feature is working as it should
- **End-to-end tests**- these tests simulate a user experience to ensure that real users have a smooth, bug-free experience.
- **Acceptance tests**- these verify the behaviour of the software under significant load to ensure its stability and reliability.

CI/CD Pipeline

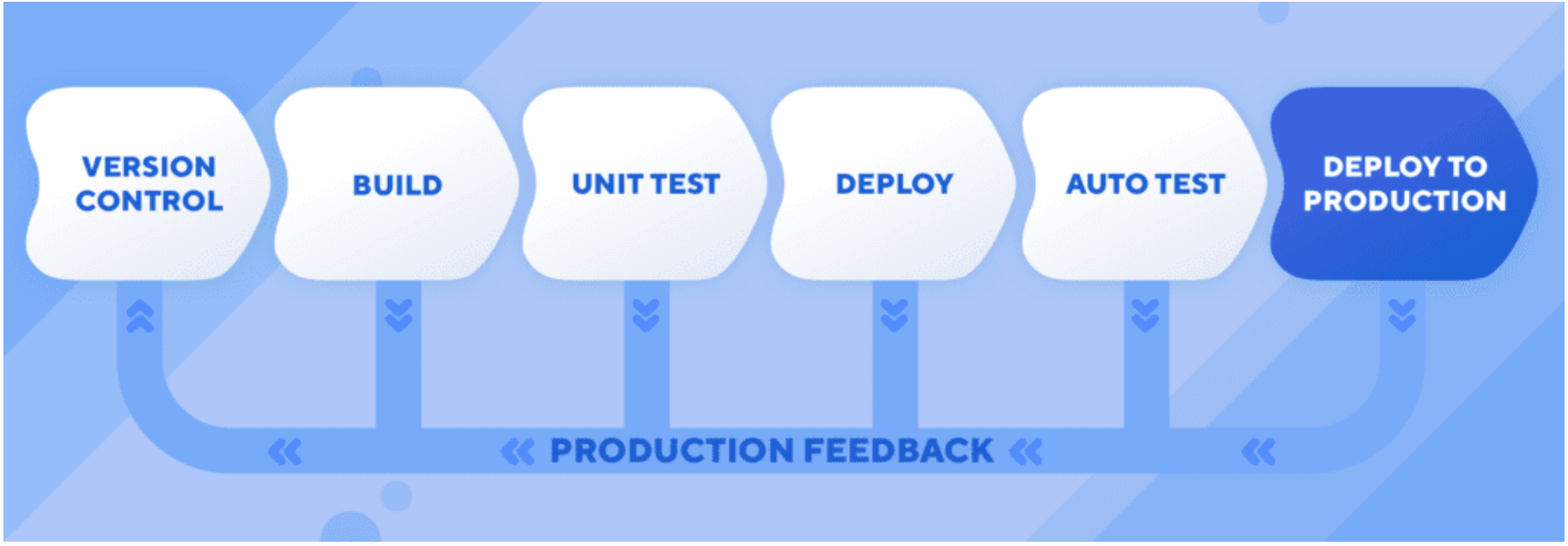


- **CI/CD pipeline:** a series of steps that should be performed to deliver a new version of the software.
- CI/CD pipelines are focused on improving software delivery via automation.
- A typical pipeline builds the code, runs tests, and then deploys the new software into production in a true replica of the software development lifecycle.
- **Building**, merging then testing the code-continuous integration
- **Preparing** the code for delivery- continuous delivery
- **Deploying** the code automatically- continuous deployment

Pipeline's Engine and Stages

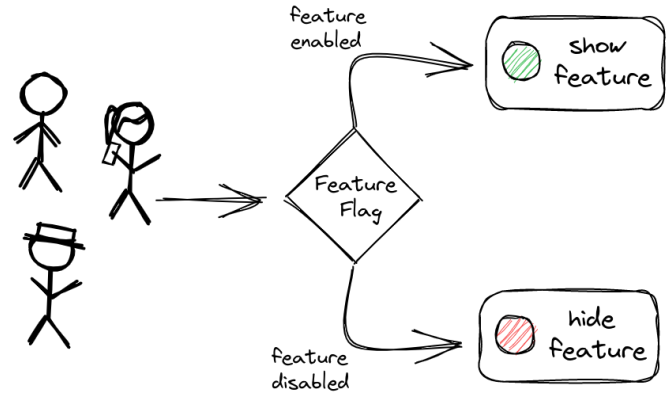
- **Source**: the CI/CD pipeline is triggered when a new code is committed to the repository.
- **Build**: this is where developers put their new code changes and compile them so they may pass through the initial testing phase
- **Test**: this is when the new code is tested through automated tests (for example, running unit tests through continuous integration). Depending on the size and complexity of the software, this step could last from seconds to hours. This stage will provide the feedback necessary for developers to fix any issues.
- **Deploy**: this is when the code is deployed to a testing or staging environment to prepare it for final release i.e continuous delivery. Usually, the build will automatically deploy once it passes through a series of automated tests.
- **Deploy to production**: here the code is released into a live production environment to reach end-users, either manually or automatically

Pipeline Example



Ref. https://katalon.com/resources-center/blog/ci-cd-pipeline

Feature Flags



- A feature flag is a software development tool whose purpose is to turn certain functionalities **ON** or **OFF** to safely test in production by decoupling code deployment from feature release.
- With feature flags, developers can push their changes without waiting for other developers by simply turning OFF the incomplete portions of the code.
- Incomplete changes can be hidden behind a feature flag while the finished changes can be released. Once the incomplete is complete, they can be turned **ON** to become visible to end-users.
- This is important as the whole aim of continuous integration is to integrate changes at least once a day, so **feature flags help maintain the momentum of continuous integration.**

Challenges & Quick Thoughts

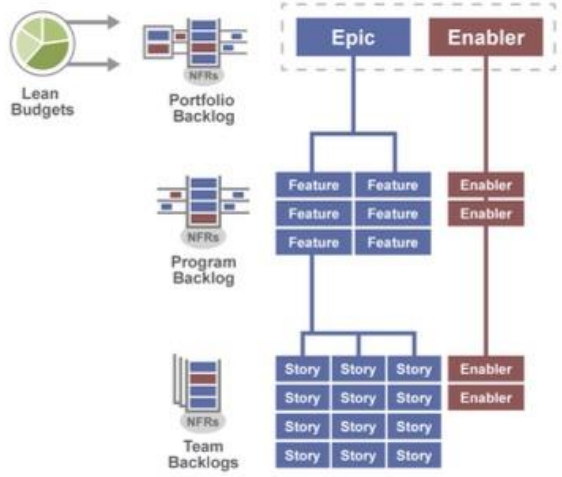
- The term “feature” in agile methodologies and CI/CD does not exist in ISO26262
- What should the feature be mapped to (e.g., function, requirement, unit, component, etc.)?
- How to fit the CI/CD into the V-model? For example:
  - Shall CI/CD be limited to SW development, or should it cover the entire V-model?
- How to construct and maintain Safety Cases in the CI/CD pipeline?
  - Continuous Safety Assurance (CSA). Maintain already existing items of evidence and highlight the missing ones
  - Can we automate the evolution and maintenance of the safety case after each deployment?
- How to manage (split, group, categorize, and prioritize) the features in the backlog? Based on:
  - their dependencies?
  - Deliveries?
  - Change containment and susceptibility to change?
  - Safety case or evidence modularity
  - Limitation by suppliers?
  - ASIL?

The Feature!

Define Features for the Program Backlog

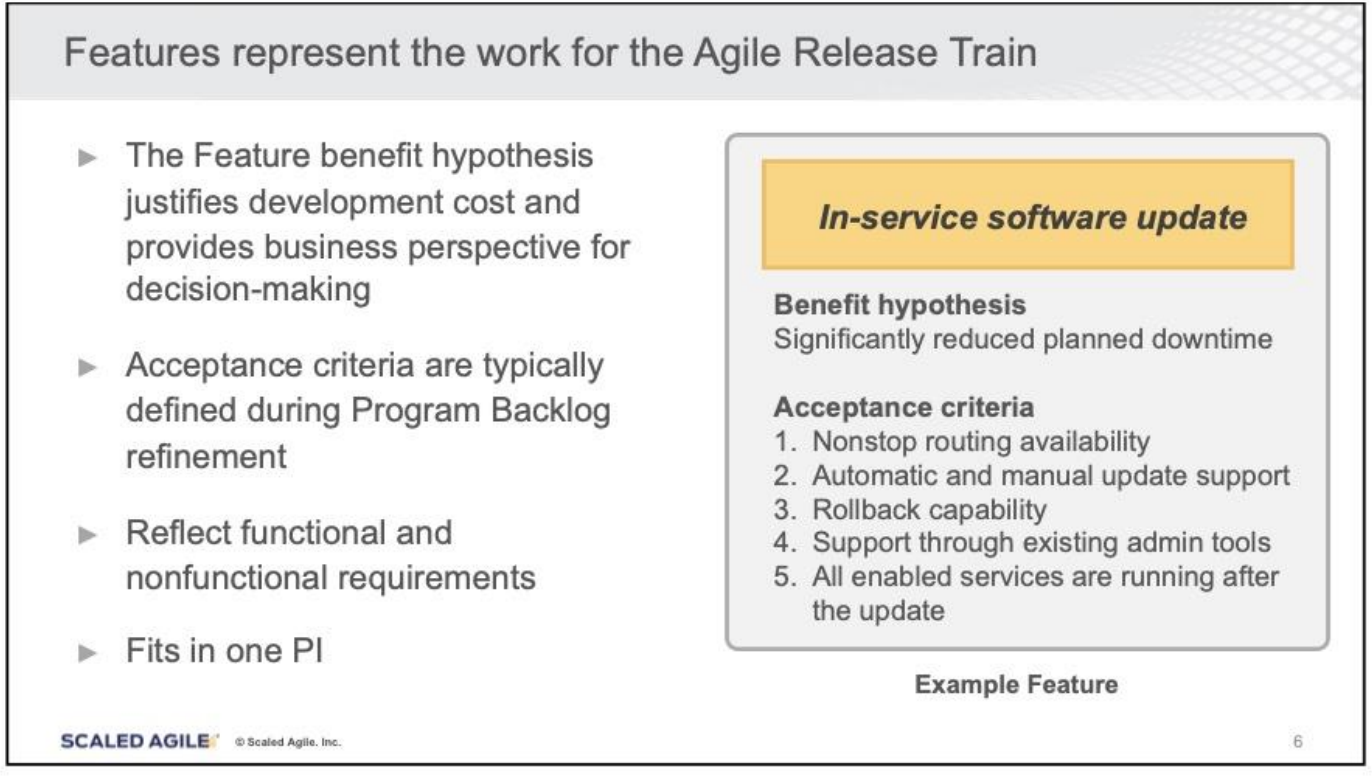
Features are services that fulfill user needs.

- Feature is an industry-standard term familiar to marketing and Product Management
- Features are described with a short phrase and a benefits hypothesis, that clearly expresses their value
- Features are identified, prioritized, estimated, and maintained in the Program Backlog



SCALED AGILE © Scaled Agile, Inc.

Example Feature

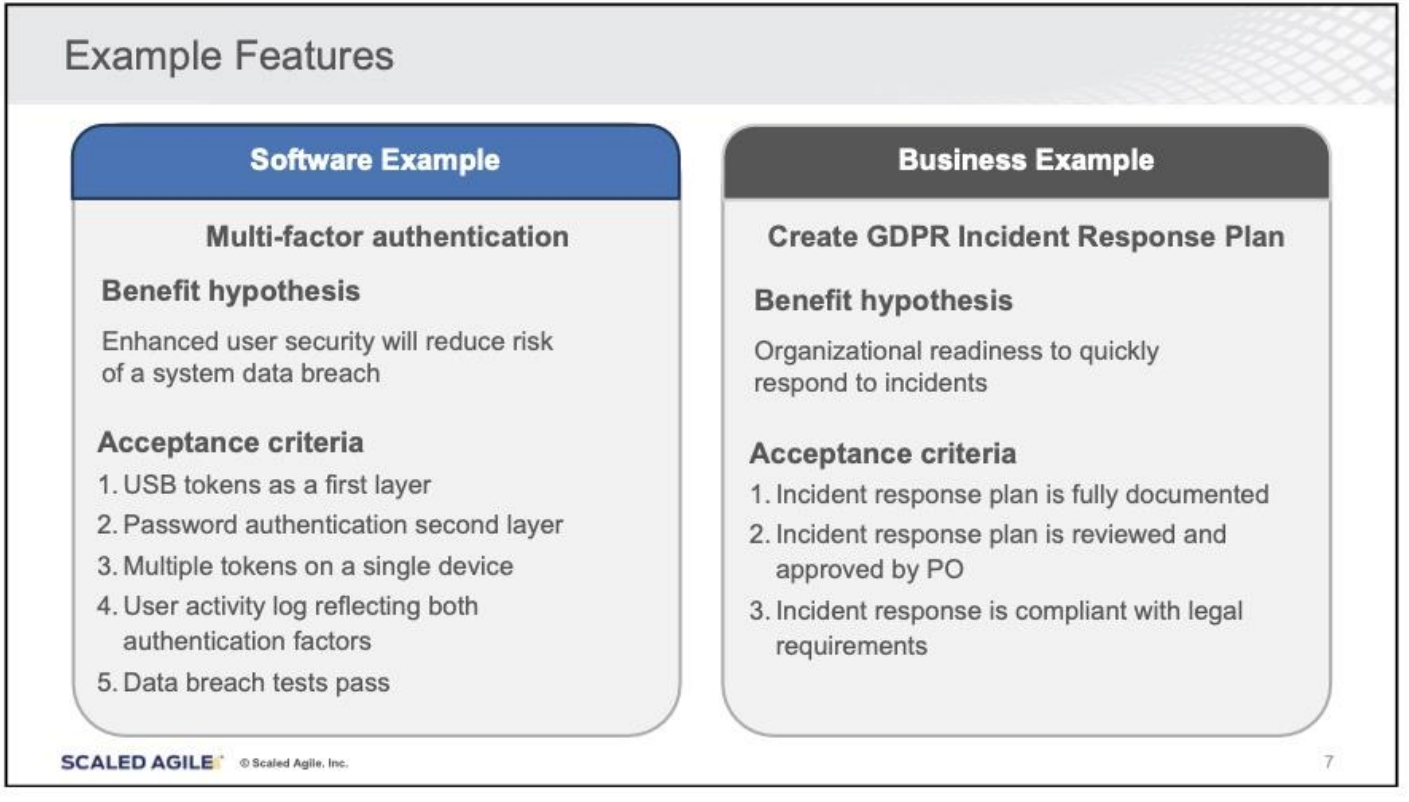


SALIENCE4CAV

Safety Lifecycle Enabling Continuous Deployment for Connected Automated Vehicles

RI. SE

Example Feature



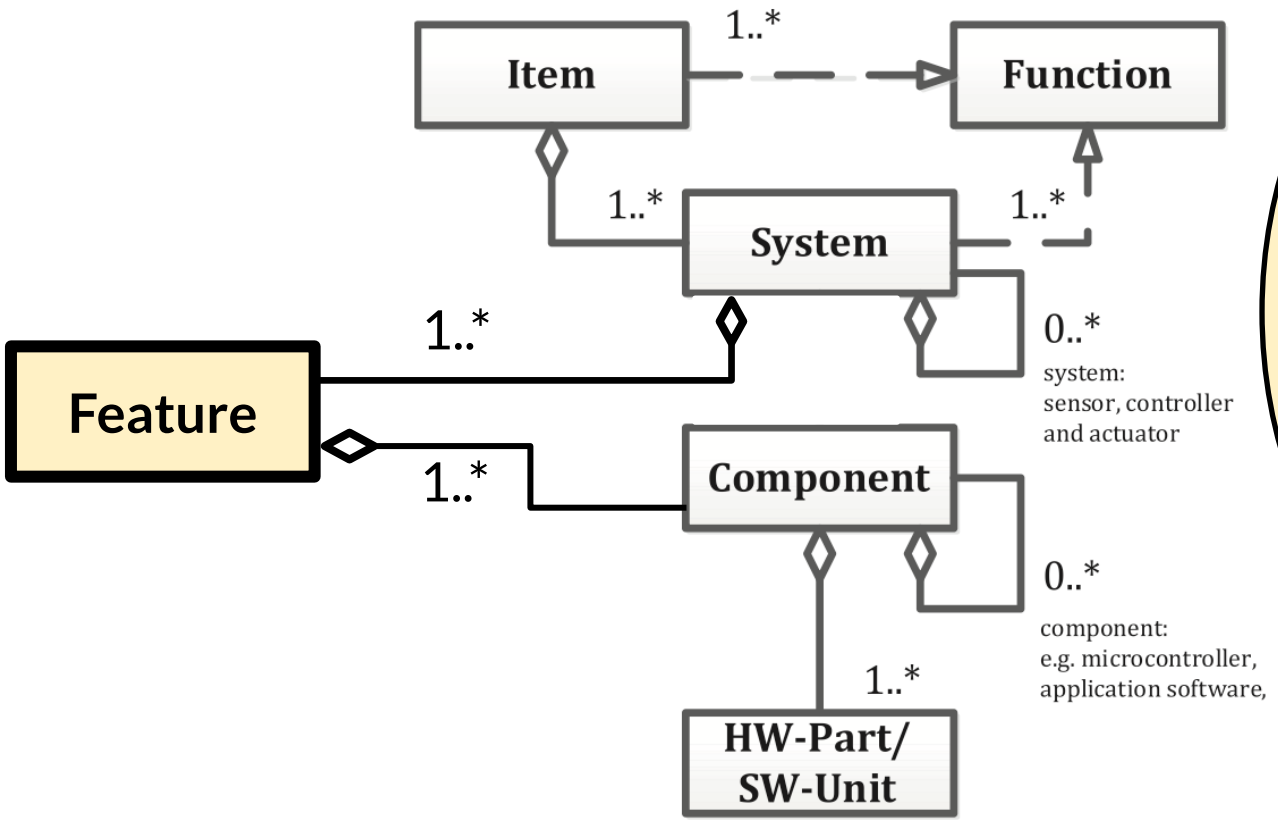
Function Vs. Item Vs. Feature

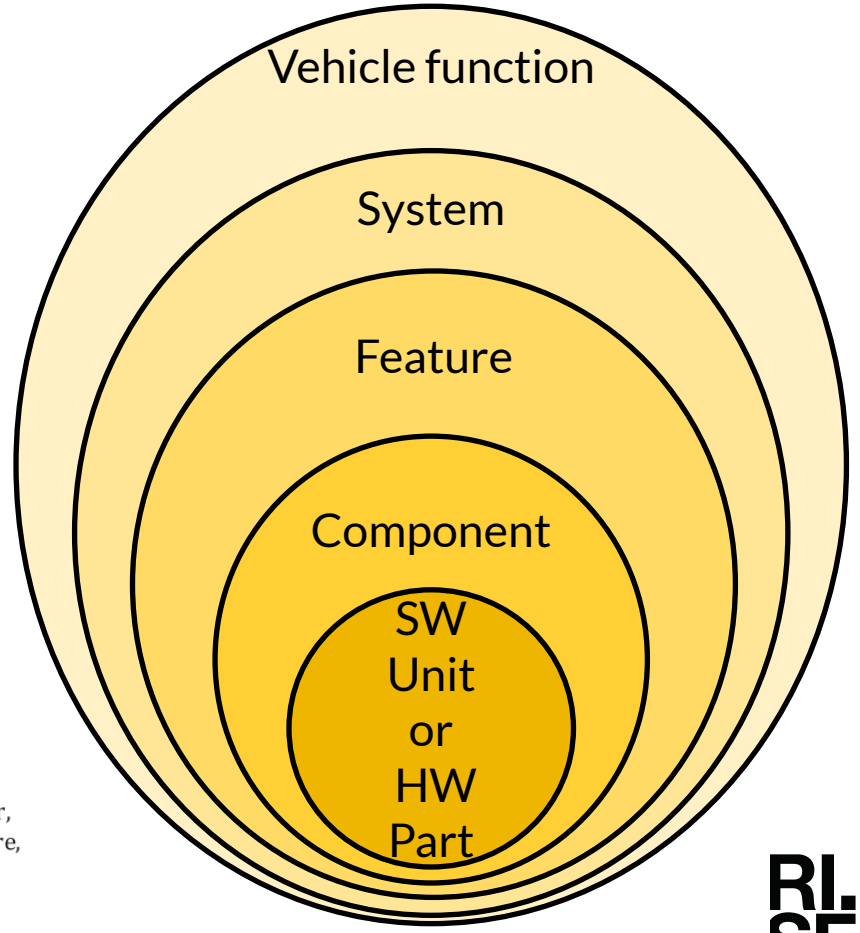
- Vehicle function (ISO26262): behaviour of the vehicle (intended by the implementation of one or more items) that is observable by the customer e.g., Autonomous Emergency Brake (AEB)
- Item (ISO26262): System or (combination of systems) that implements a function or part of a function at the vehicle level e.g., AEB can be an item
- In agile methodology, a feature is a service or function of the product that delivers business value and fulfills the customer's need. Each feature is broken down into several user stories, as it is usually too big to be worked on directly
- Fitting the the agile's def. of the feature into ISO 26262 context so that it is *a building block for a system or systems that implement item(s)*

    e.g., the brake pedal position is a feature that contributes to accomplishing vehicle functions such as braking

- Hint: CI/CD features can be inspired and derived from the HAZOP functions list

Specific proposal





RI.
SE

Thank you!

SALIENCE4CAV

Safety Lifecycle Enabling Continuous Deployment for Connected Automated Vehicles

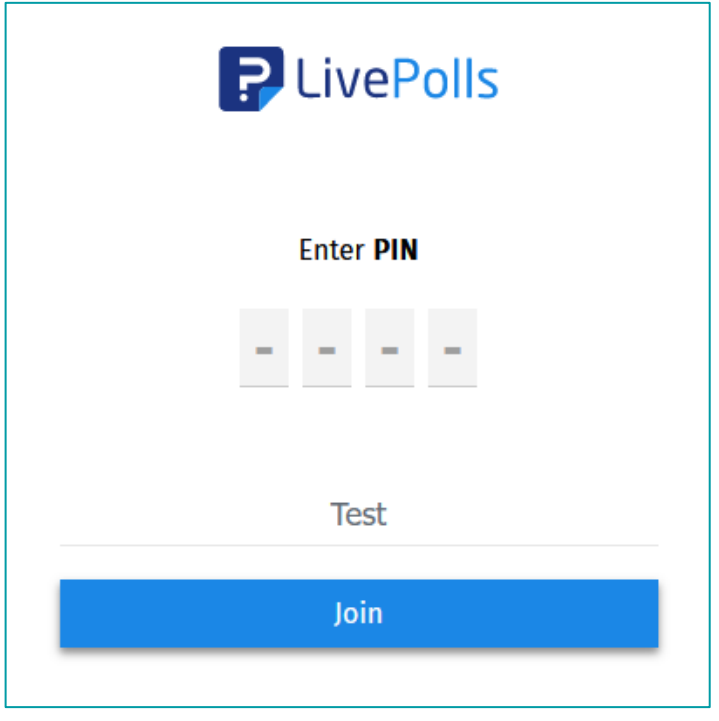
RI.SE

Questions for world café discussions

1. How to manage and reconcile the impacts of product-line variability, necessary changes and unexpected side effects?
2. What parts of the development life cycle needs to be considered in a CI/CD pipeline to support safety assurance?
3. How would consistency between changes in safety requirements, architecture, implementation and different variants be assured in a CI/CD tool chain?
4. How to continuously maintain the safety case evidence after a system change or increment?
5. How can the safety case be projected to highlight its updated parts after a system change efficiently?
6. How the safety claims can be validated against the new safety boundaries or thresholds?
7. How do you think CI/CD works with current safety standards and regulations?

https://www.questionpro.io/

1. Do you think that CI/CD is more capable of reproducing the deliverables and all safety-related verification and validation work products than the traditional WoW? [yes, no, can't say]
2. Do you think it is feasible to identify and analyse potential failure modes (that system changes might frequently introduce) by following CI/CD WoW?
3. Do you believe that CI/CD shall consider system boundaries and requirements freeze at some point during the development lifecycle?



Questions

Continuous Assurance and Contract-Based Design

SALIENCE4CAV

Safety Lifecycle Enabling Continuous Deployment for Connected Automated Vehicles

RI.SE

SALIENCE4CAV

Safety Lifecycle Enabling Continuous Deployment for Connected Automated Vehicles

Safety assurance & Safety contracts

in Continuous Deployment

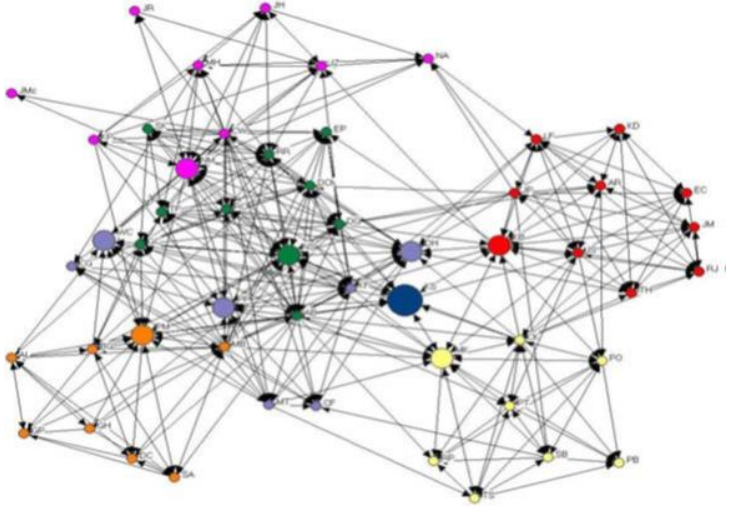
Anders Cassel

Date: 2022-11-23

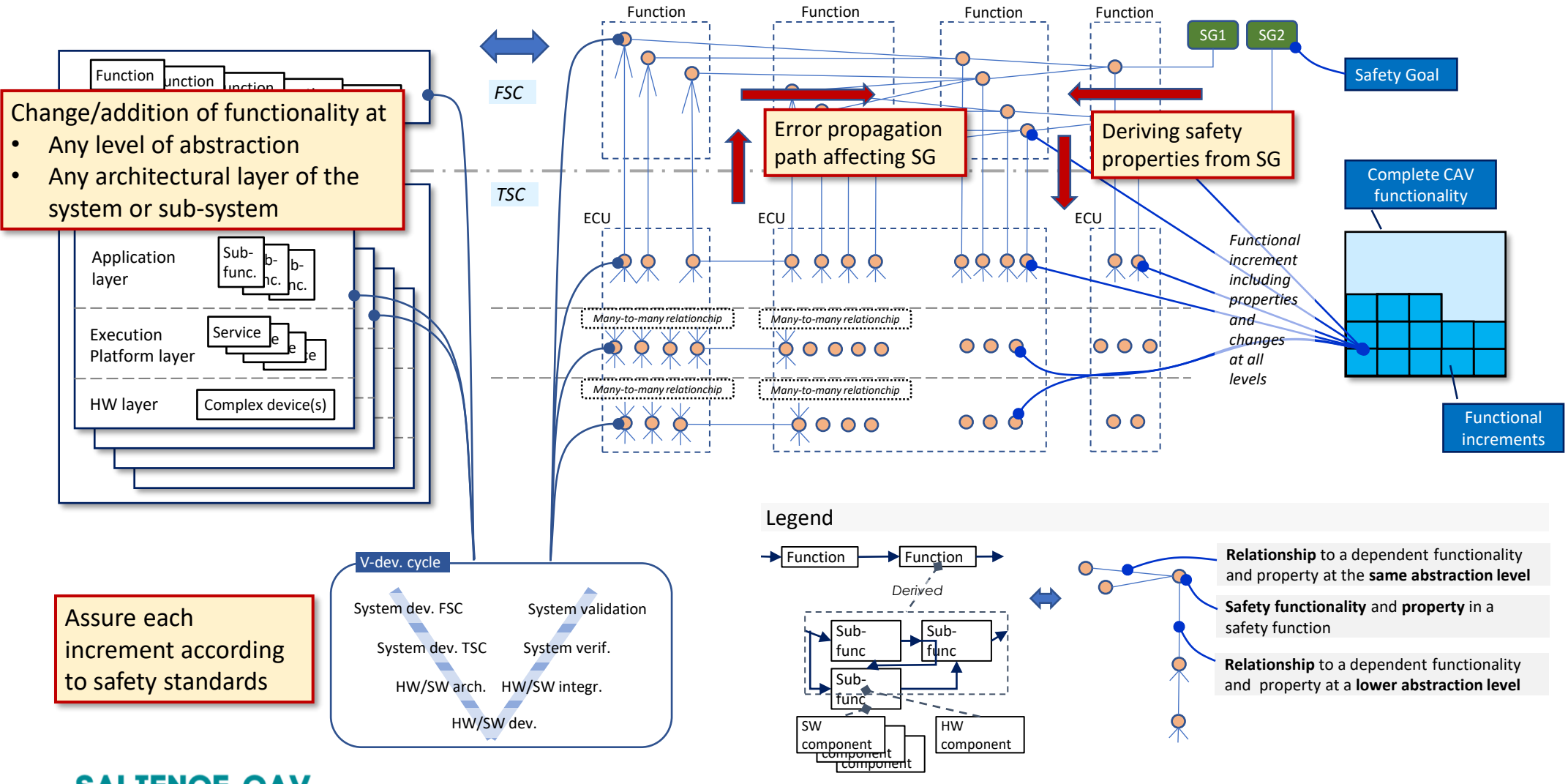
SALIENCE4CAV qamcom

How to master a complex world of

- Agile development methods & simultaneous engineering
- Central compute architectures
- Frequent system release cycles
- Conform to safety & security standards

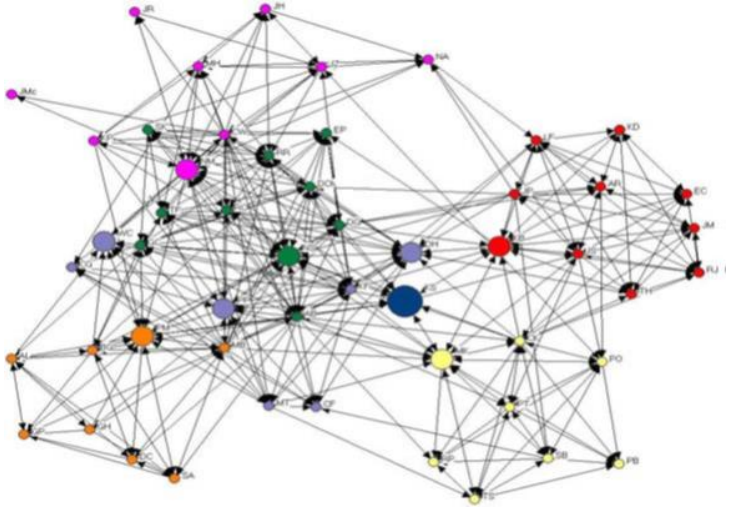


Continuous deployment of ADS in agile development – A complex reality

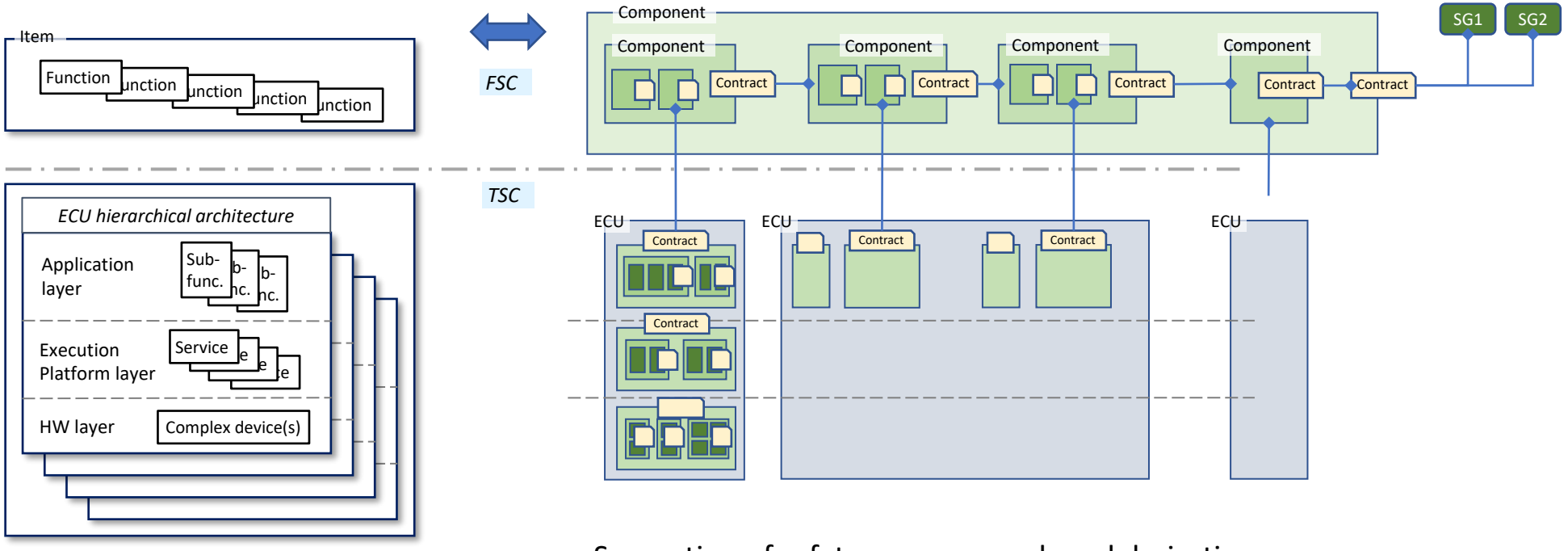


SALIENCE4CAV

How to master the complexity



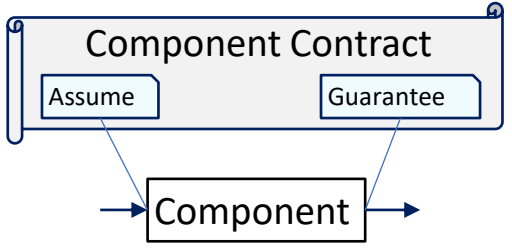
Mastering complexity – Introduction of Contract-based design



- Separation of safety concerns and modularization
  - Component-based design
  - Hierarchy of components → highest Item level to atomic level
- Safety properties and functionality assured by Safety contracts
  - Component safety property and functional response guaranteed by Safety contract
  - Contracts specified for each component at all abstraction levels
  - Higher level contract assured by fulfillment of lower-level contracts

- Safety contracts
  - Methodology focusing on separation of concerns
  - Assuring safety properties and behavior of each component
  - Expressed by formal requirement syntax
  - Enables automatic contract checking
  - Safety contracts part of architecture model at all design levels by e.g. SysML/UML, EAST-ADL,..
  - Safety case compilation based on safety contracts by SW tool support integrated in CI/CD build chain

- Component Assume-Guarantee (A/G) Contracts
  - **Contracts** are defined as **Assume-Guarantee** assertion pairs
  - **Guarantee** are the guaranteed functionality that the specific component is able to fulfill.
  - **Assume** are interpreted as a set of assumptions on the signals provided at their input-ports and the operational environment required for the component functionality.
  - Component response and properties are guaranteed under a set of assumptions on the environment, e.g. inputs and dependencies
  - **Top-down** & **bottom-up**
    - **Global properties** of systems are composed based on local properties of the components
    - **Local properties** of components are decomposed based on properties at a higher abstraction level



Types of Contracts

Component Contract

- **Pair of assertions** of **assumptions** and **guarantees** of a specific component

Component-Component Interface Contract

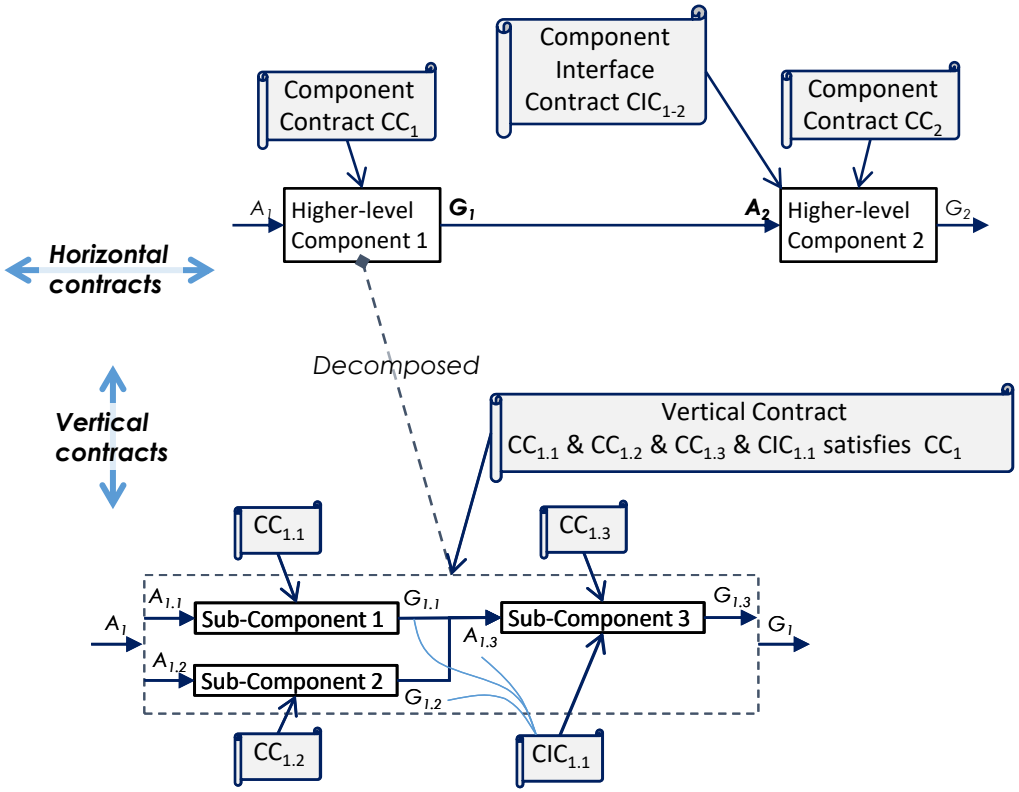
- Relationship between assumptions of a specific component and guarantees of the interfacing components.
- The guarantees of a component output-port must satisfy the assumptions of the signal input-port of the receiving component(s).

Horizontal Contracts

- Component Contracts and Component Interface Contracts defined at the same abstraction level.

Vertical Contracts

- Decomposition of a higher-level component contract to a set of sub-component contracts and component interface contracts
- Sub-component contracts satisfies the higher-level component.

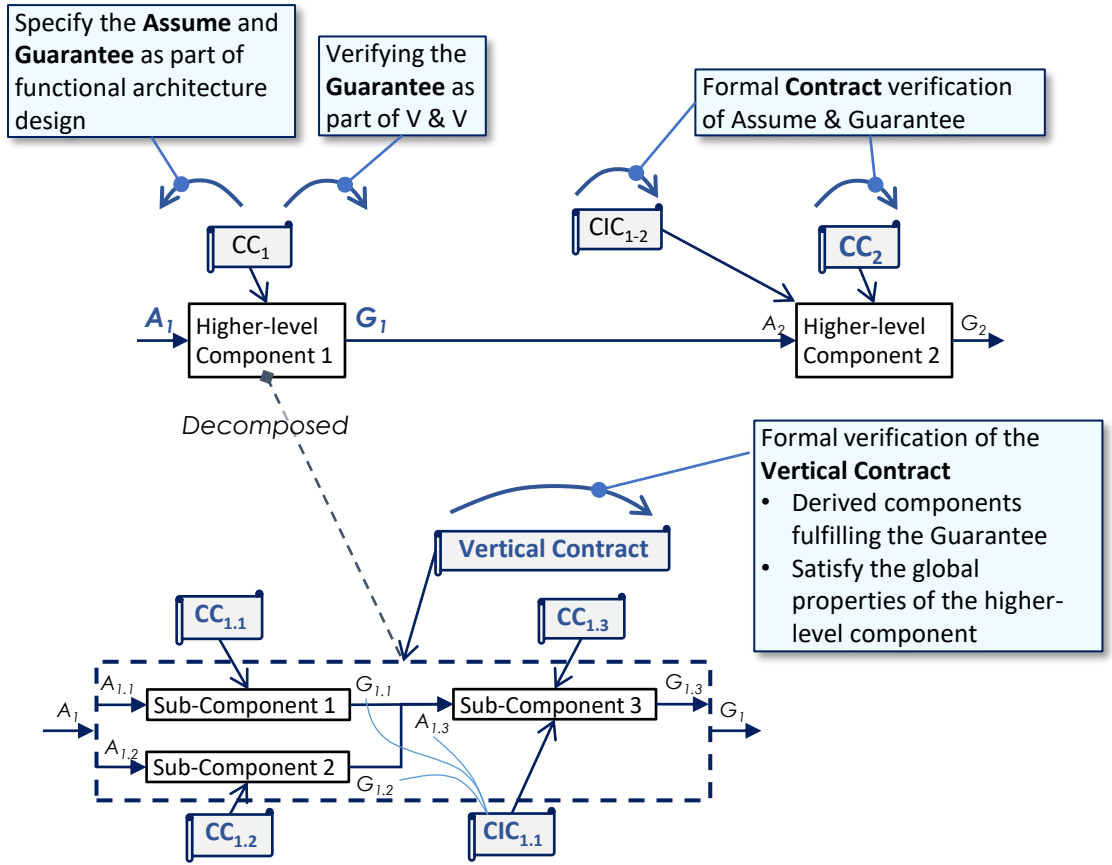


Formal checking of contracts

- Contracts are specified by a defined requirement syntax implementing a set of logic expressions
- Result of verification activities feedback into the contract model

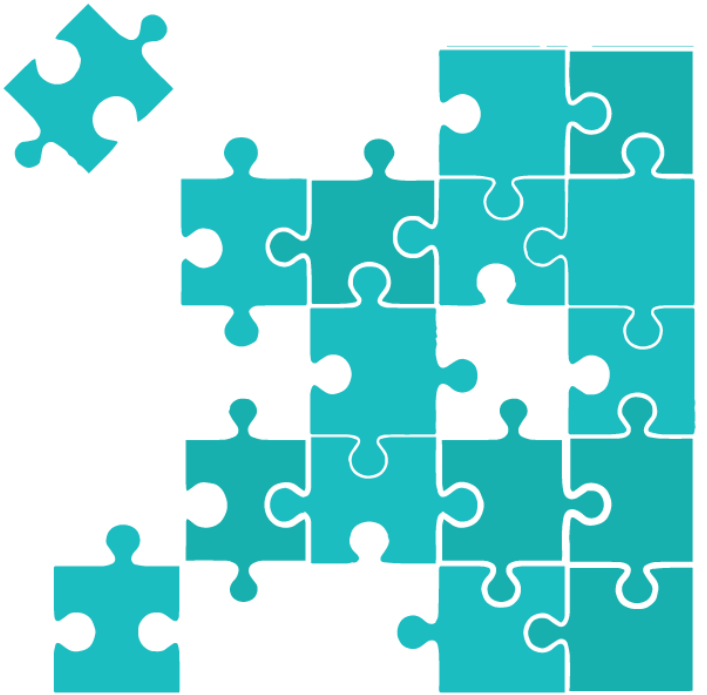
- Component contracts are verified that the guarantee is realized and satisfies the assumption

- Formal verification:
  - Checking the formal assertions and verification result of assume and guarantee
  - **Pass**: Behavior and properties of assume and guarantee meets the criteria
  - **Fail**: Behavior and properties of assume and guarantee don't meet the criteria
  - Supports impact and variability analysis



Fitting components & contracts together

- Meta-models

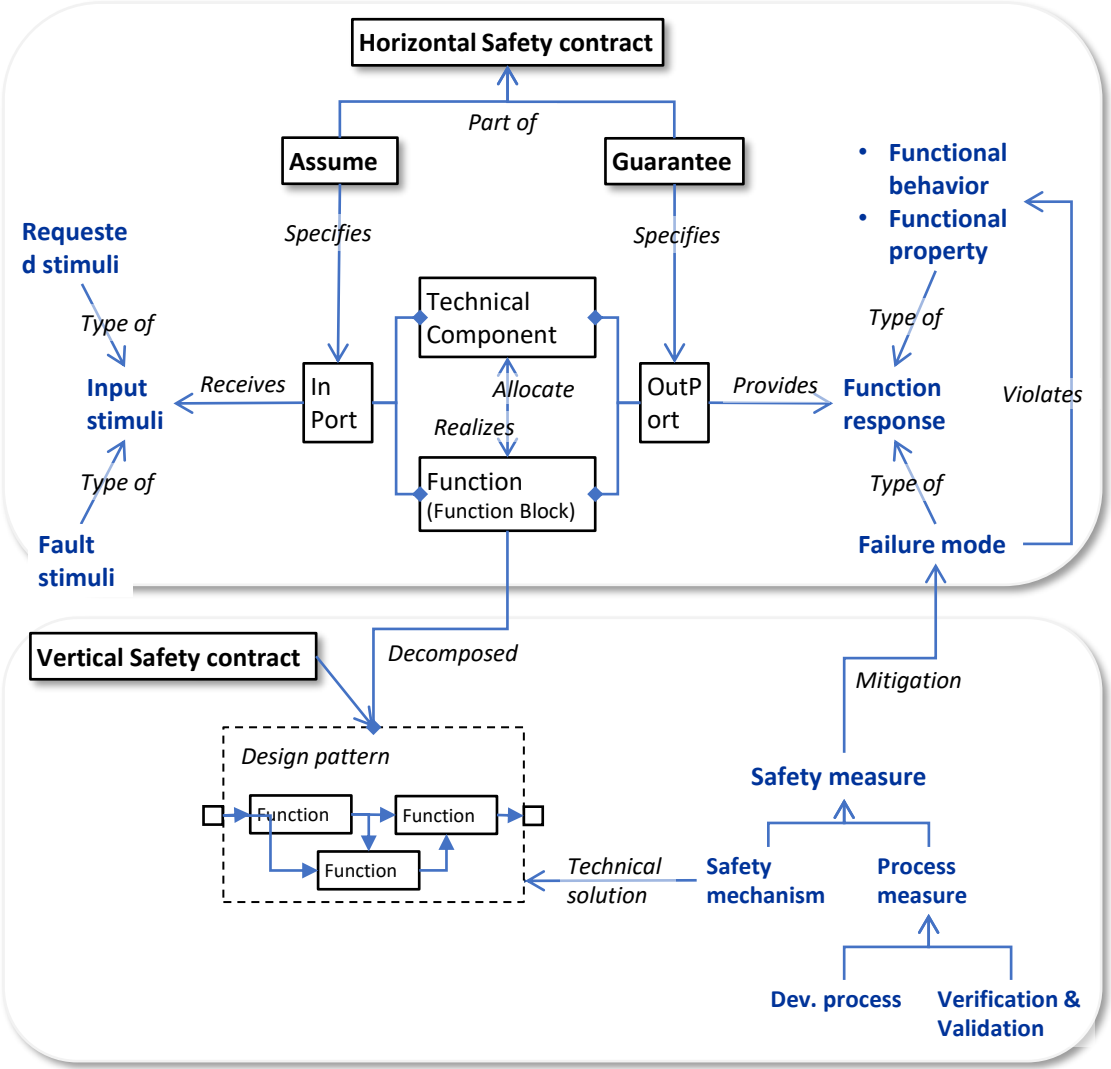


Safety contract model - Simplified

Vertical Contract

Expression of

- Safety properties
- Fault & Error propagation affecting safety properties
- Safety mechanisms
- The Guarantee
  - Decomposed properties corresponds with higher-level contract
  - Detection & Prevention of error propagation
- The Assume
  - Decomposed input signals corresponds with higher-level contract
- Process measures
  - Validation & Verification, e.g. test cases
  - Evidence of Validation & Verification result



Horizontal Contract

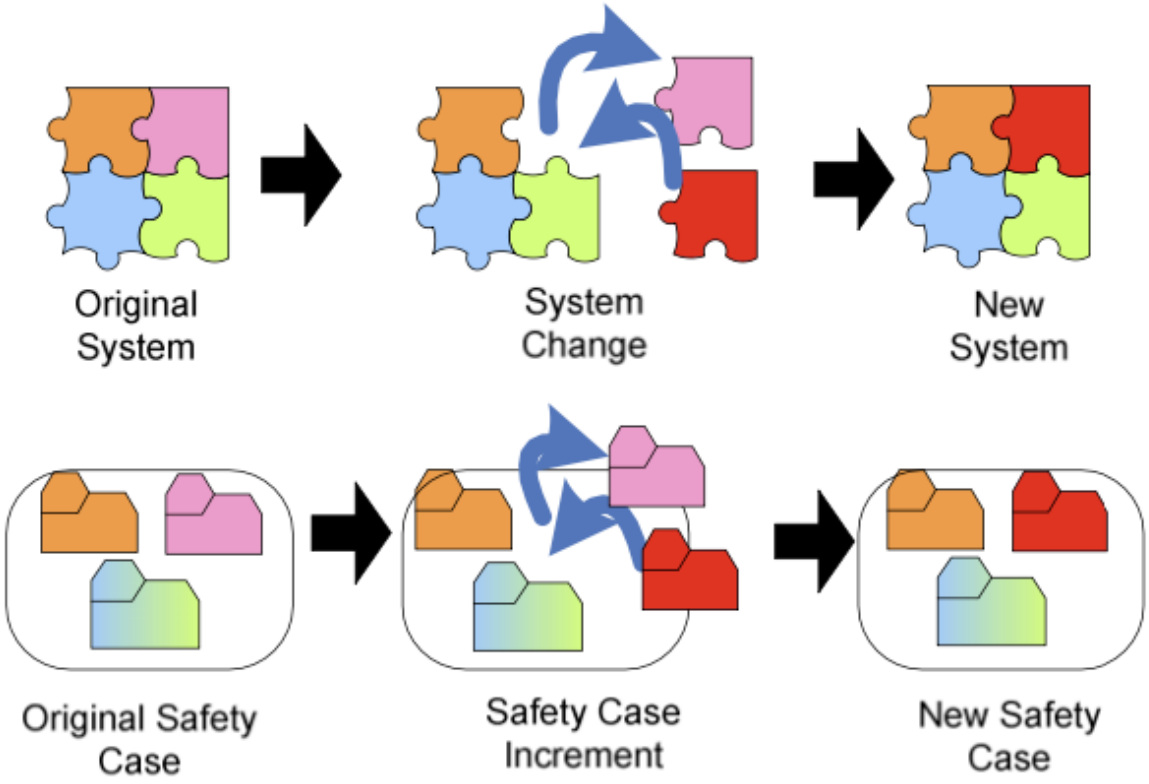
Expression of

- Safety properties
- Functional behaviour
- Function response
- Safety relevant failure mode
- The Guarantee
  - Function response is within safety limits
  - Response if error is present
  - Probability of failure mode
- The assume
  - Input signals
  - Operational environment
- Process measures
  - Validation & Verification, e.g. test cases
  - Evidence of Validation & Verification result

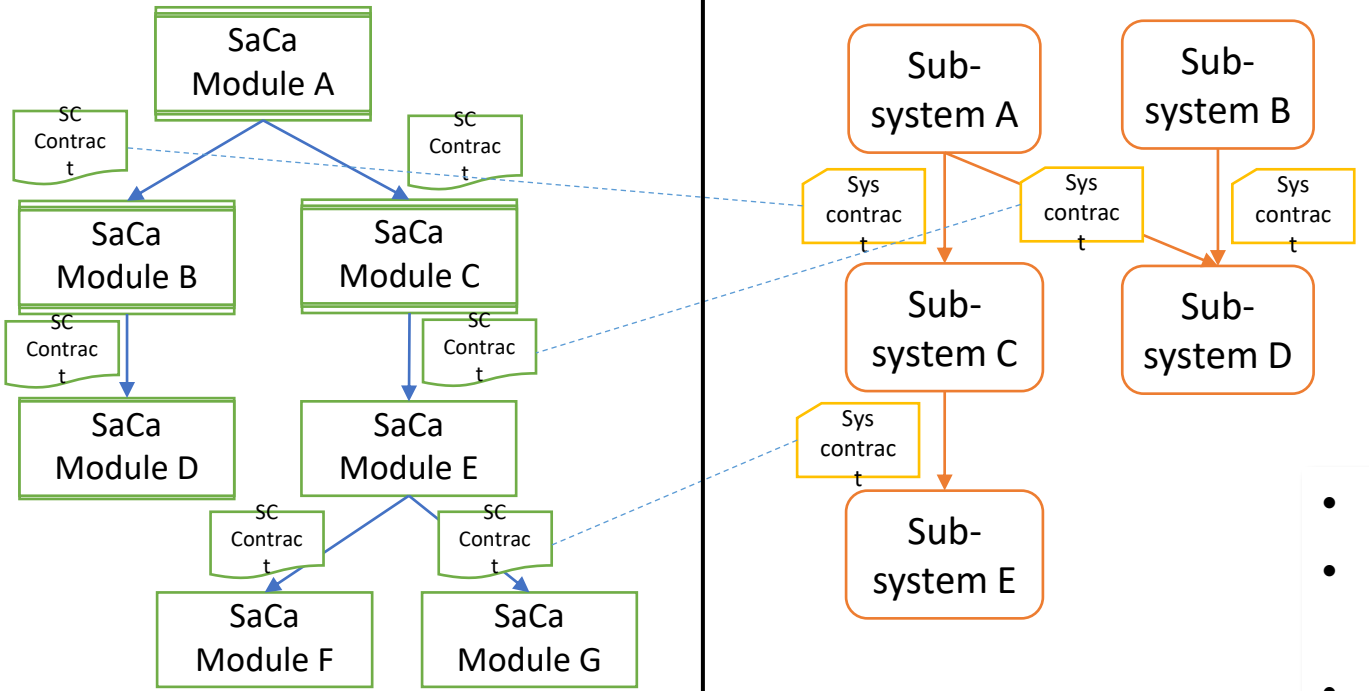
SALIENCE4CAV

Safety case model

Change in a system component corresponds with a safety case increment

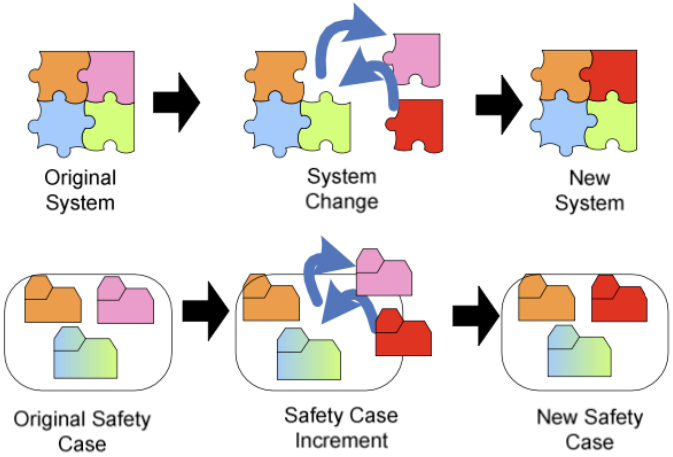


Safety case model



Safety case domain

System domain



- Safety cases are modular
- Each component has a set of safety contracts
- Safety verification can be limited to the incremental change and its impact on the complete system

SALIENCE4CAV

- **Component-based design** to enable separation of concerns, re-use of components and usage of safety contracts
- **Safety contracts** to assure safety functionality and properties of safety relevant components
- **Verification measures** and results are part of safety contracts
- **Formal methods** for automatic consistency check of safety contracts
- **Defined safety case argumentation structure** and verification criteria for safety case compilation
- **Automated analysis** integrated in the CI/CD tool chain
  - Variability analysis
  - Error propagation analysis
  - Impact and deviation analysis

- Contracts for system design (A Benveniste, INRIA 2012)
- Assurance aware contract-based design for safety-critical systems (I Sljivo, 2018)
- AMASS research project - Baseline and requirements for architecture-driven assurance (AMASS_D3.1_WP3_FBK_V1. 1, 2018)
- AMASS research project - Design of the AMASS tools and methods for architecture driven assurance (AMASS_D3.3_WP3_INT_V1.0, 2018)
- Continuous Deployment for Dependable Systems with Continuous Assurance Cases (F. Warg, et al., 2019)

End

Thank you

Anders Cassel
Anders.cassel@qamcom.se

qamcom

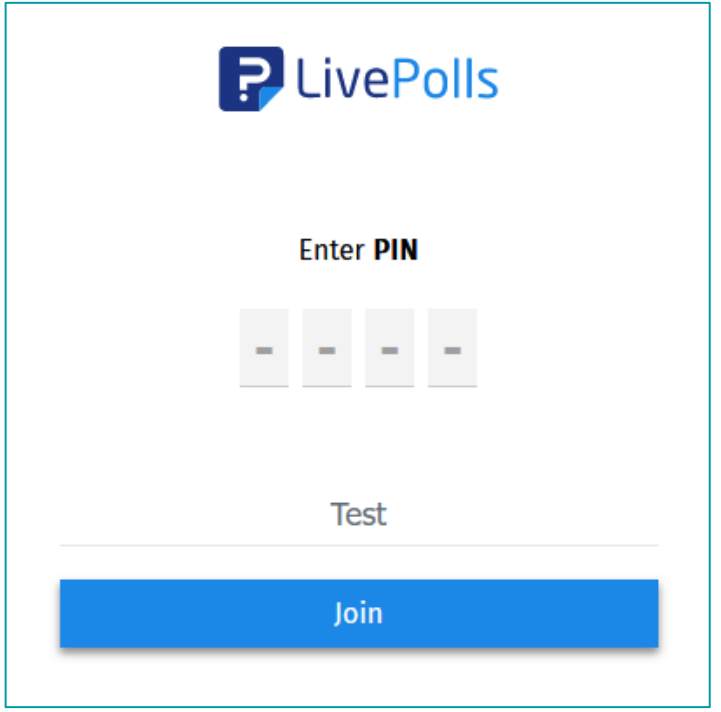
Qamcom Research & Technology

Questions for world café discussions

1. What things in safety-contract methodology are preventing this methodology to be used in practice?
2. If suitable tools would exists, how would then safety-contracts benefit different development organisations at different abstraction levels?
3. What in the current development methods are most difficult to master when practicing agile development and simultaneous engineering at several abstraction levels?
4. How would safety assurance measures (e.g. audits, assessments etc) benefit from safety-contract based design?
5. What challenges are there to apply component-based design for ADS at all abstraction levels, i.e. from Item level to atomic SW- and HW-component level?
6. How to derive safety contracts systematically (e.g. driven by safety analysis outcomes)?
7. How to measure and ensure the completeness, correctness, and consistency of safety contracts?
8. How to continuously check that the safety contracts capture the guarantees of the desired safe performance?

https://www.questionpro.io/

1. (Background) What is your primary role in your organisation?
2. (Background) What is your organisation primary business?
3. Did you ever hear of safety contracts and their usage in safety-critical system engineering before today?
4. Would safety-contract based design benefit your organisation (e.g. negotiating requirements and changes between organizations)?
5. Would you consider using contract-based design if suitable tools exists?



Questions

Contact the workshop organizers

- *Project Manager* - Dr. Fredrik Warg, RISE Research Institutes of Sweden, email: fredrik.warg@ri.se
- *Researcher* - Anders Cassel, Qamcom Research &Technology, email: anders.cassel@qamcom.se
- *Researcher* - Dr. Anders Thorsén, RISE Research Institutes of Sweden, email: anders.thorsen@ri.se
- *Researcher* - Dr. Omar Jaradat, Qamcom Research &Technology, email: omar.jaradat@qamcom.se
- *Researcher* - Dr. Negin Nejad, Qamcom Research &Technology, email: negin.nejad@qamcom.se
- *Researcher* - Prof. Dejiu Chen, KTH, email: chendj@kth.se
- *Researcher* - Stig Ursing, Semcon, email: Stig.Ursing@semcon.com
- *Researcher* - Victoria Vu, Semcon, email: victoria.vu@semcon.com

More information

http://salience4cav.se/

Dr. Fredrik Warg
fredrik.warg@ri.se
RISE Research Institutes of Sweden

Image by Arek Socha from Pixabay