ZENZIC

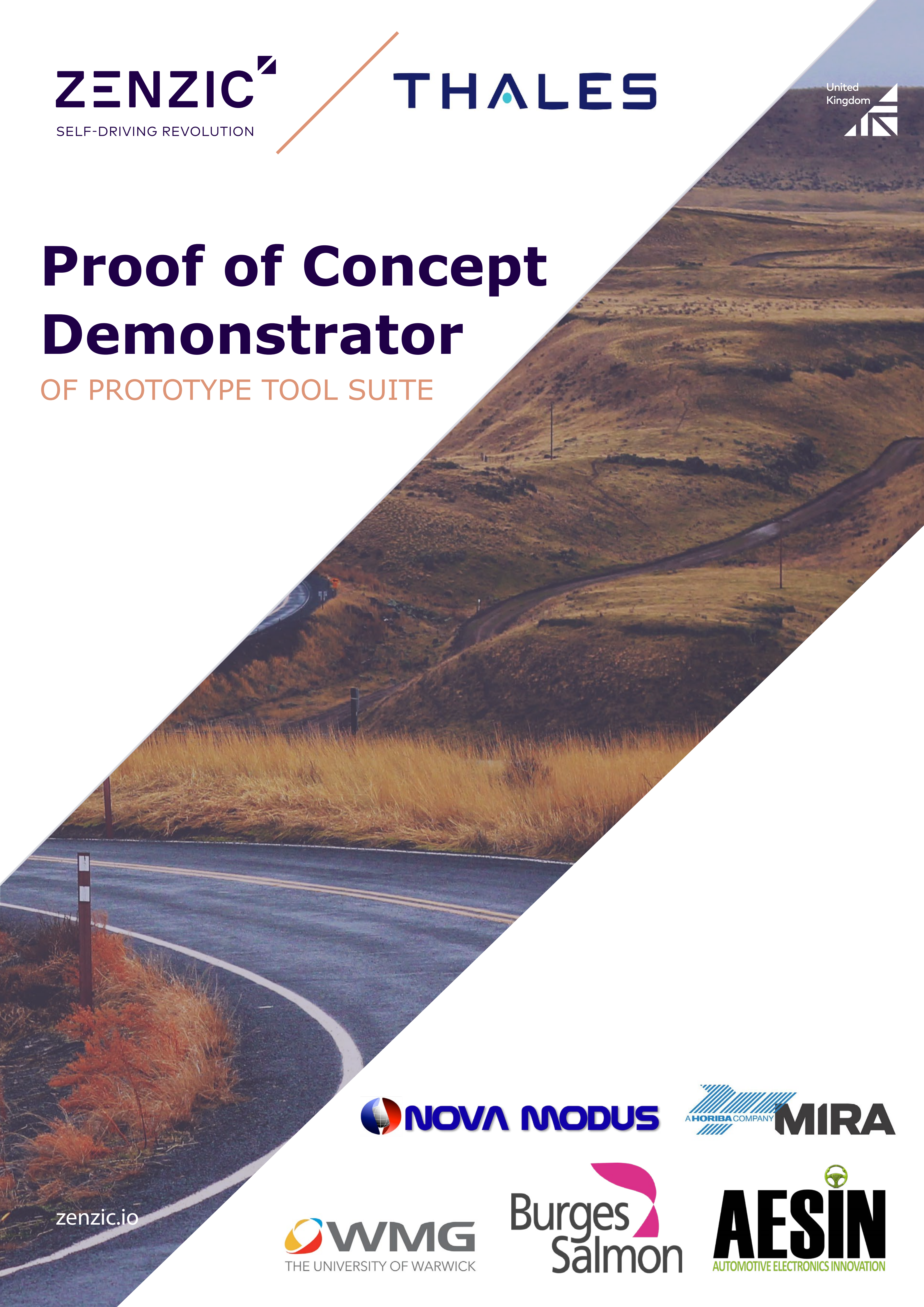
SELF-DRIVING REVOLUTION

THALES

United Kingdom

Proof of Concept Demonstrator

OF PROTOTYPE TOOL SUITE



zenzic.io

NOVA MODUS

A HORIBA COMPANY MIRA

WMG THE UNIVERSITY OF WARWICK

Burges Salmon

AESIN AUTOMOTIVE ELECTRONICS INNOVATION

THALES UK LIMITED

350 Longwater Avenue Reading RG2 6GF
Tel. +44 0)118943 4500
www.thalesgroup.com

The ResiCAV+ contributors would like to acknowledge

ZENZIC
SELF-DRIVING REVOLUTION

Centre for Connected & Autonomous Vehicles

Innovate UK

in making this study possible.

About Us

Thales - Together. Safer. Everywhere

Thales plays a role whenever critical decisions need to be undertaken. In all the markets we serve – aerospace, space, ground transportation, security and defence – our understanding of the Critical Decision Chain helps customers to decide and act in a timely fashion and obtain the best outcomes.

World-class technologies and the combined expertise of 65,000 employees in 56 locally based country operations make Thales a key player in assuring the security of citizens, infrastructure and nations.

Thales UK Limited, Research and Technology

Thales UK's Reading-based research and technology facility is the UK arm of the Thales corporate research centre. Activities focus on providing solutions: Security and Communication Systems, Galileo and Position-Based Systems and Enhanced Digital Environments. These are based on the key technologies of IP Networks and Network Security, Wireless Communications, Sensors and Signal Processing, and Navigation and Positioning. The facility offers a wide range of consultancy and development services to European Government Agencies and to industry throughout the world.

Executive Summary

The goal of ResiCAV is to explore the technological and economic feasibility of developing, implementing and operating a sustainable UK Cybersecurity Engineering capability; to ensure the cyber resilience of future mobility.

This is an ambitious goal considering the size and complexity of the issue, but at the same time a very necessary target when we consider the possible risk of catastrophic failure in moving from Connected and Autonomous Vehicle (CAV) demonstrations to mass deployment if new methods are not developed to protect, detect, understand and react to emerging threats.

Thus, the electronic systems of future vehicles, as well as the intelligent transport systems that they interact with, will need to exhibit a high degree of resilience to a wide range of threats.

ResiCAV+ documents this study in 5 Reports:

- **Report 1: Prototype Tool Suite** (software/documentation/skills training course?) including worked Examples of their use to provide cyber resilience in real-world automotive systems but noting OEM/Tier1 specifics may need to be redacted from a final report.
- **Report 2: Proof of Concept Demonstrator of Prototype Tool Suite** a report on the tool suite covering how and in what form the tool suite might be made available for use by commercial and academic researchers, allowing them to start to integrate the CyRes methodology into existing practices.
- **Report 3: Legal Report** providing a route to a per vehicle **legally defensible** argument that the cyber vulnerability of the braking system was reduced ALARP using this ***significant difference*** approach and based on the distributed ledger.
- **Report 4: Compliance Report** providing a route to a per vehicle providing a route to a per vehicle real time compliance argument based on the use of the distributed ledger.
- **Report 5: Exploitation Plan** an updated roadmap outlining next steps to improve cyber-resilience of automotive systems using CyRes tools & methodologies, building on previous work by team, ResiCAV, NCSC, etc.
  - Roadmap for continuing development and the funding required.
  - Plans to exploit and disseminate tools and methods on a global basis.

This report (Deliverable 2) outlines the Proof of Concept Demonstrator of Prototype Tool Suite covering how and in what form the tool suite might be made available for use by commercial and academic researchers, allowing them to start to integrate the CyRes (Cyber Resilience) methodology into existing practices. The goal of CyRes is to define an operational methodology, suitable for standardisation, for which:

1. **The methodology itself** is capable of being tested in court or by publicly appointed regulators.
2. **Operators** understand what evidence should be produced by it and are able to measure the quality of that evidence.

3. **The evidence produced** is capable of being tested in court or by publicly appointed regulators.

Typically this will mean that the real-world system to which the methodology has been applied is capable of operating at all times and in all places with a legally acceptable value of negative consequence.

However, and unlike other Cyber Security methods for which the question is often ‘how much must I spend to be compliant’, the CyRes methodology is rooted in **economic advantage** with achieving the conditions for compliance being a by-product. One of the fundamental insights of CyRes is that **Cyber-attacks are ‘emergent properties’.** Much of the **value of the economy** is based on products and services created around emergent properties so by understanding and managing these in real time CyRes for the first time allows Cyber Resilience to sit firmly within the value creation rather than compliance chain.

The ResiCav+ programme, and the research by leading experts that underpins it, builds on prior work that demonstrated not only is the CyRes methodology is economically and technologically feasible but that operating in this way would allow the **direction of capital towards growth rather than compliance**. Under this programme

CyRes achieves this dynamic business advantage whilst at the same time developing evidence of Cyber Resilience that can be confidently brought to court if required. It is built around three principles and six certification arguments designed to provide **a mathematically well-founded index of resilience, including cyber resilience, in operational space**.

In considering the technological and economic feasibility of CyRes this study has concluded that all of the **resilience and economic benefits** are achievable by the UK in **3 years** subject to an **investment of £150m**. This conclusion is made on the basis that:

1. All of the elements necessary to operate CyRes exist today at a Technological Readiness Level (TRL) of 5-7.
2. Subject to an appropriate investment in tools of less than £20m over three years then the method could achieve TRL 9 and would be globally economically attractive.
3. Within three years the method could deliver a level of resilience with respect to emergent properties and cyber-attacks that would exceed current state of the art.

Whilst it was not the main purpose of this study the research conducted determined that **more than 25% of the cost of vehicles** was being spent in chips, software, V&V and compliance / type approval aimed at removing inherent diversity and turning them back into entities with inherent and increasing potential for global catastrophic outcomes; **much of this could be saved by using CyRes**.

Furthermore, collaborators at all levels of the supply chain reported that up to **5% of the cost of the digital vehicle** was being spent on Cyber Security with **little or no measurable outcomes** with respect to Cyber Resilience.

The techniques demonstrated by ResiCav+, which support the CyRes methodology and its operationalisation, provide an **academically well-founded baseline for understanding the cost of operation** together with an **academically well-founded baseline for understanding the effectiveness**. The acceptance of CyRes and this baseline together with the need to understand and improve it will provide a **springboard for innovation** for companies at all layers together with a **direction for and measure of academic research** in the coming decades.

The purpose of the demonstration testbed was to:

- Support development of the CyRes capability

- Allow systems, services and applications e.g. CAVs to be integrated together
- Support development and demonstration of CyRes scalability
- Demonstrate the production of court admissible evidence to support the CyRes arguments.

To enable the testbed to meet this purpose, the following requirements had to be met:

- Modular integration capability, such that inclusion of a new component does not impact other components
- Include a distributed ledger as a system of systems data store and allows systems to interact
- Provide an evidential chain of information
- Provide interfaces to allow systems, services, applications and components to integrate into the system of systems
- Enable simulators or real systems to be integrated for development, demonstration and experimental purposes
- Possess visual interfaces to demonstrate the CyRes concepts and methods.

Contents

Table of Figures

1. Introduction

The purpose of the demonstration testbed was to:

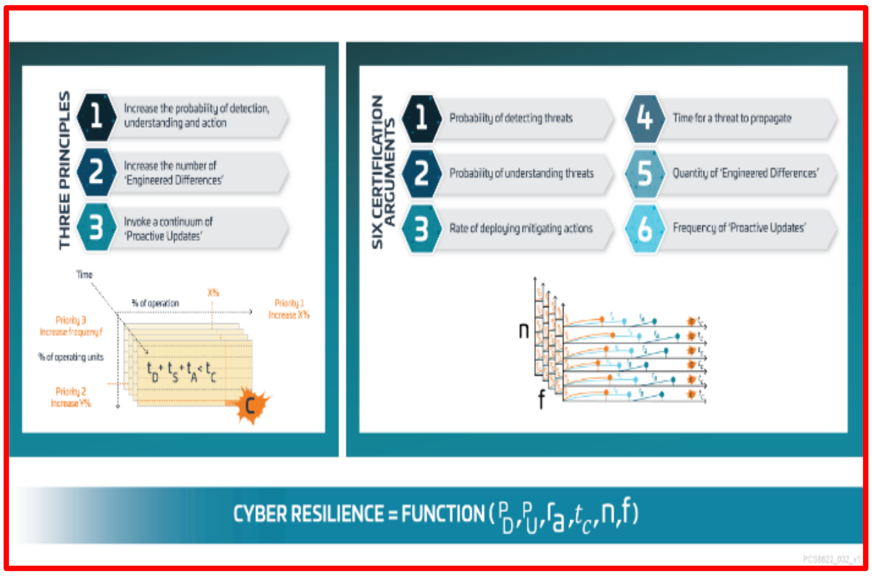
- Support development of the CyRes capability
- Allow systems, services and applications e.g. CAVs to be integrated together
- Support development and demonstration of CyRes scalability
- Demonstrate the production of court admissible evidence to support the CyRes arguments.

To enable the testbed to meet this purpose, the following requirements had to be met:

- Modular integration capability, such that inclusion of a new component does not impact other components
- Include a distributed ledger as a system of systems data store and allows systems to interact
- Provide an evidential chain of information
- Provide interfaces to allow systems, services, applications and components to integrate into the system of systems
- Enable simulators or real systems to be integrated for development, demonstration and experimental purposes
- Possess visual interfaces to demonstrate the CyRes concepts and methods.

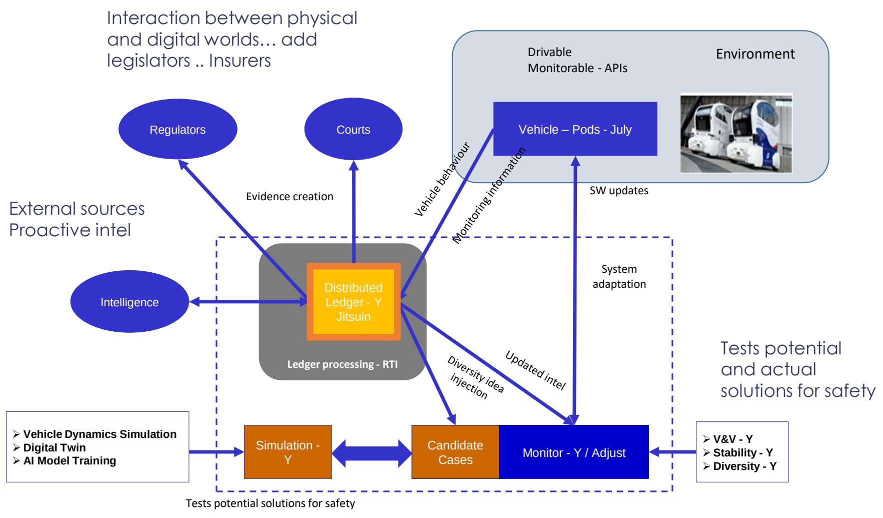
2. Document Structure

Section 3: The Problem – based on previous research articulates the problem space for modern and future vehicles including CAVs including the 3 principles and 6 arguments that the CyRes methodology identified as fundamental to a coherent and defensible resilient system.



THREE PRINCIPLES
1 Increase the probability of detection, understanding and action
2 Increase the number of 'Engineered Differences'
3 Invoke a continuum of 'Proactive Updates'
Time
% of operation
X%
Priority 1 Increase X%
Priority 3 Increase frequency f
% of operating units
Priority 2 Increase Y%
$t_D + t_S + t_A < t_C$
C
SIX CERTIFICATION ARGUMENTS
1 Probability of detecting threats
2 Probability of understanding threats
3 Rate of deploying mitigating actions
4 Time for a threat to propagate
5 Quantity of 'Engineered Differences'
6 Frequency of 'Proactive Updates'
n
f
CYBER RESILIENCE = FUNCTION (P_D, P_U, r_a, t_C, n, f)

Section 4 The ResiCav+ Technical Approach – The ResiCAV+ technical approach sets out a feedback loop based around the collection of evidence relating to events and the decisions made with respect to those events that provides for the reinsertion of tested updates into vehicles at scale and within minutes. It operates on the basis that changes to the vehicle baseline must often be driven by changes necessitated by emergent properties in its environment rather than from controlled software updates. As such and given that the Technology Readiness Level (TRL) of an emergent property will often be 0 this toolset was created on the basis that Automotive systems will at all times be in a design phase crucially even when deployed.



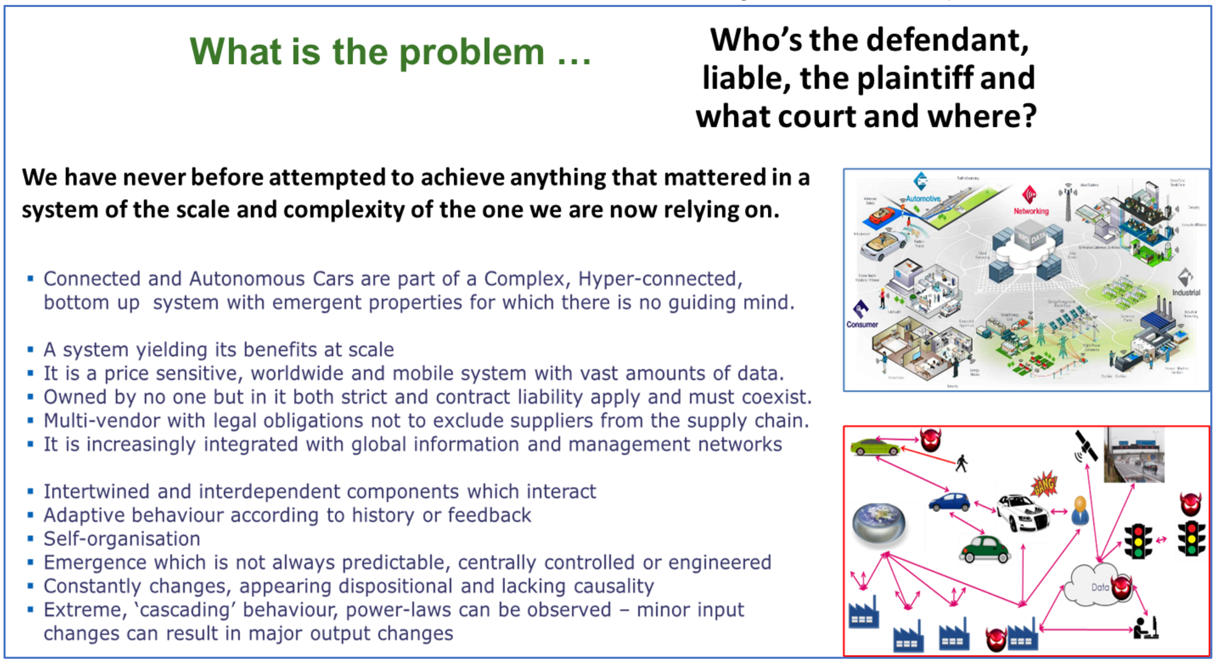
Section 5 Some Conclusions and Further Work – This section documents some of the conclusions from the ResiCAV+ study particularly with respect to the tools and their suitability for use.

Appendices – The Appendices are as follows:

- Appendix A sets out some initial work that has been done to create standardised schema and 'smart contracts' for use when communicating between 3rd party tools and the blockchain framework.
- Appendices B to F cover the various tools that were integrated together into the ResiCAV+ Framework to show the feasibility of a legally sustainable real time capability to cover the type of complex system that the automotive industry is now evolving to.
- Appendix G covers screenshots from RKVST which was the blockchain tool used in this demonstration
- Appendix H covers elements from the Braking demonstration.

3. The Problem

Over the period since 2015 we have characterised every significant emerging system as having all, or at least most, of the characteristics shown below under the diagram ‘what is the problem’



What is the problem …

Who’s the defendant, liable, the plaintiff and what court and where?

We have never before attempted to achieve anything that mattered in a system of the scale and complexity of the one we are now relying on.

- Connected and Autonomous Cars are part of a Complex, Hyper-connected, bottom up system with emergent properties for which there is no guiding mind.

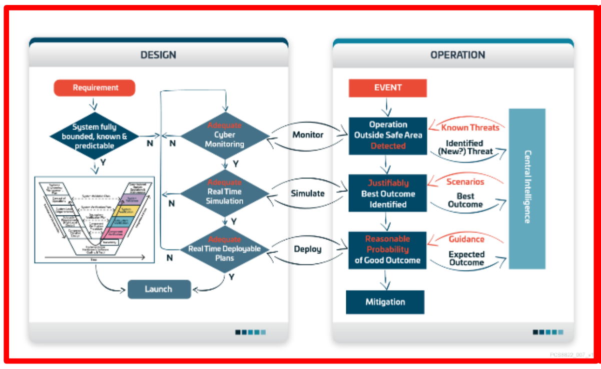
- A system yielding its benefits at scale
- It is a price sensitive, worldwide and mobile system with vast amounts of data.
- Owned by no one but in it both strict and contract liability apply and must coexist.
- Multi-vendor with legal obligations not to exclude suppliers from the supply chain.
- It is increasingly integrated with global information and management networks

- Intertwined and interdependent components which interact
- Adaptive behaviour according to history or feedback
- Self-organisation
- Emergence which is not always predictable, centrally controlled or engineered
- Constantly changes, appearing dispositional and lacking causality
- Extreme, ‘cascading’ behaviour, power-laws can be observed – minor input changes can result in major output changes

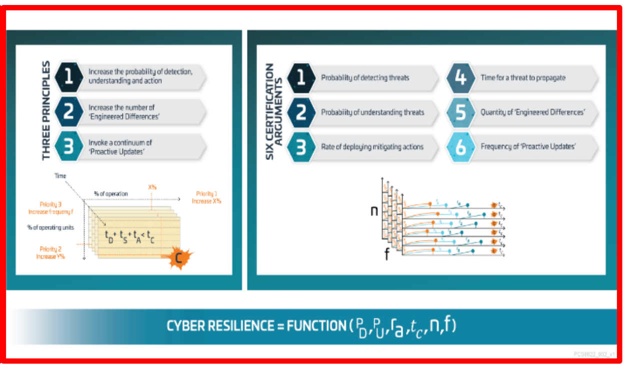
These systems can be stimulated to failure by cyber attacks. Whilst demonstrating the problem at the level of the global interconnected system we have also demonstrated it in secure chipsets and in systems including automotive braking, flight and energy control systems; all areas where the highest level of safety might be expected to apply and for which we are now demonstrating that the very engineering process being used for assurance are in fact the major contributor to the most catastrophic failures.

This problem is increasingly acknowledged and cited by C level business leaders and those of their supply chains, particularly in areas including Automotive and Medical where safety of life is a consideration. Evidence of this can be seen in the recent ministerial response to a Techworks letter with respect to the Automotive Transformation Fund and the Connected and Automated Mobility (CAM) Technology Acceleration Fund where the only subject explicitly now called out is Cyber Security.

In response to this problem and further observations with respect to complexity science, the nature of



DESIGN
Requirement
System fully bounded, known & predictable
Adequate Cyber Monitoring
Adequate Real Time Simulation
Adequate Real Time Deployable Plans
Launch
Y
N
Monitor
Simulate
Deploy
OPERATION
EVENT
Operation Outside Safe Area Detected
Justifiably Best Outcome Identified
Reasonable Probability of Good Outcome
Mitigation
Known Threats
Identified (New?) Threat
Scenarios
Best Outcome
Guidance
Expected Outcome
Central Intelligence



THREE PRINCIPLES
1 Increase the probability of detection, understanding and action
2 Increase the number of ‘Engineered Differences’
3 Invoke a continuum of ‘Proactive Updates’
Time
% of operation
X%
Priority 1 Increase X%
Priority 3 Increase frequency f
% of operating units
Priority 2 Increase Y%
$t_D + t_S + t_A < t_C$
C
SIX CERTIFICATION ARGUMENTS
1 Probability of detecting threats
2 Probability of understanding threats
3 Rate of deploying mitigating actions
4 Time for a threat to propagate
5 Quantity of ‘Engineered Differences’
6 Frequency of ‘Proactive Updates’
n
f
CYBER RESILIENCE = FUNCTION (P_D, P_U, r_a, t_c, n, f)

evidence and significant work has been done to define a method based on 3 principles and 6 arguments to move the development of assurance to the operational rather than design space.

In electro-mechanical systems, each system and platform is different and consequently each would be expected to be susceptible to failure in a different way and at a different time; the potential ‘harm’ arising from failure occurs with statistical probability, one device at a time. It is this principle that forms the basis of standard safety calculations. In digital systems it is possible that an identified fault could manifest in all digitally identical systems at the same time, thereby giving rise to global catastrophic failure. That is, at the level of the overall system of devices rather than at the level of the individual device.

Sharman et al. (2004), proposed functionality defence by heterogeneity as a paradigm for securing systems. This technique was inspired by the biological phenomenon of the human race surviving deadly viruses because of the diversity arising from heterogeneity. We use similar inspiration to create the concept of engineered differences. The approach is concerned with the deliberate introduction of significant differences between systems and platforms. These differences may be imperceptible to the user or operator but may prevent all systems being affected in the same way by cyber-attacks. At its most extreme one may suppose that a cyber physical system in which every entity was different would fail, and therefore be subject to calculation of harm, in the same way as an electro-mechanical system.

Producing a system that exhibits significant differences in this way has been deemed economically infeasible on the basis that:

1. the need for V&V and product certification could not be borne on a per unit basis;
2. the economics of eg. chip design and manufacturing, together with the necessary ecosystem fabs, toolchains etc. favour a volume industry in which if you are not #1 or #2 then you are not likely to survive.

In addition:

3. it has been found to be technologically difficult to determine that one system or component is in fact significantly different from another.

For the engineering of significant difference to be a viable technique it must be possible to address each of the three points above.

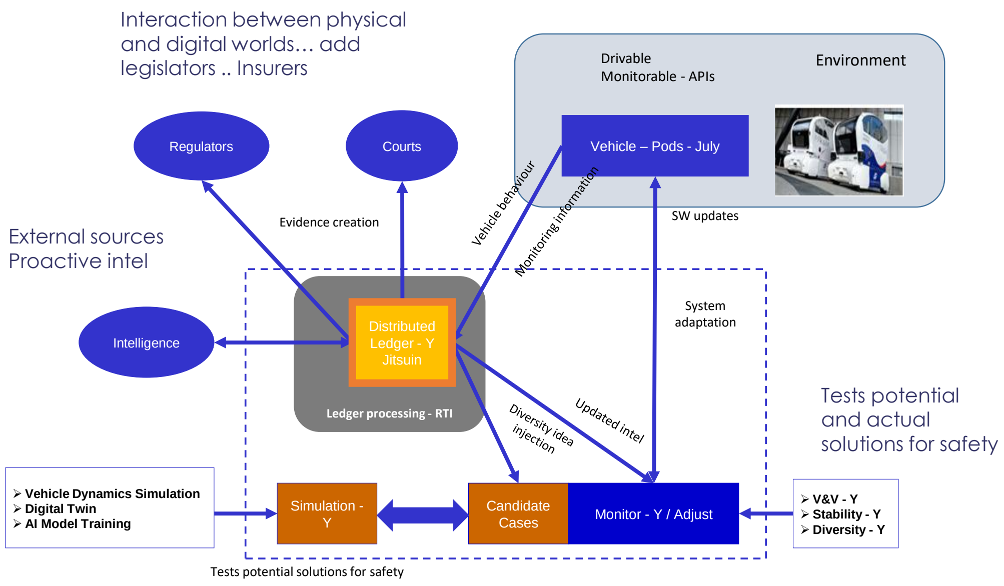
In 'Technological and Economic Feasibility of the CyRes Methodology' [Thales 2020] drawing on areas including *Approximate Computing*, *CMOS Variability Modelling and Prediction*, *Nanoelectronics Devices and Modelling*, *Significant Difference* and *Appropriate Level of Resilience and Risk Management* to demonstrate that *Significant Difference* is economically and technologically beneficial and is feasible. Further it was shown that the investment currently made to turn inherently analogue systems into digital is creating significant vectors for catastrophic failure, diminishing the anticipated associated benefits that have driven the industry of the past decades.

Resting on the work of many of the ACE-CSRs and all of the NCSC supported RIs, 'Technological and Economic Feasibility of the CyRes Methodology' did significant amounts of the scientific groundwork and suggested a set of tools which over a 3-5 year period could allow the economically feasible operationalisation of future methods to combat cyber and complexity attacks against our most important products and systems.

ResiCAV+ has set out to demonstrate a basis of an extensible framework into which those tools might be integrated and that these tools could form the basis of a legally defensible and adoptable engineering methodology for the Automotive industry throughout its supply ecosystem.

4. The ResiCav+ Technical Approach

The ResiCav+ technical approach set out to demonstrate that 'events' occurring in the vehicle due to changes in the environment in which it was operating could be detected, understood, a solution determined and reloaded into the vehicle within minutes if necessary and that this could be done in such a way that the requirements of both regulators and the law could be met. In doing so ResiCAV+ has demonstrated that CyRes can be applied to provide both system resilience and evidence for courts or regulators.



The solution uses multiple simulations of various aspects of the vehicle, against candidate uses cases, to predict how the system will behave. These use-cases and the model are bound within a set of assumptions about the range of inputs & behaviours in the deployed system – these are the constraints of the simulation.

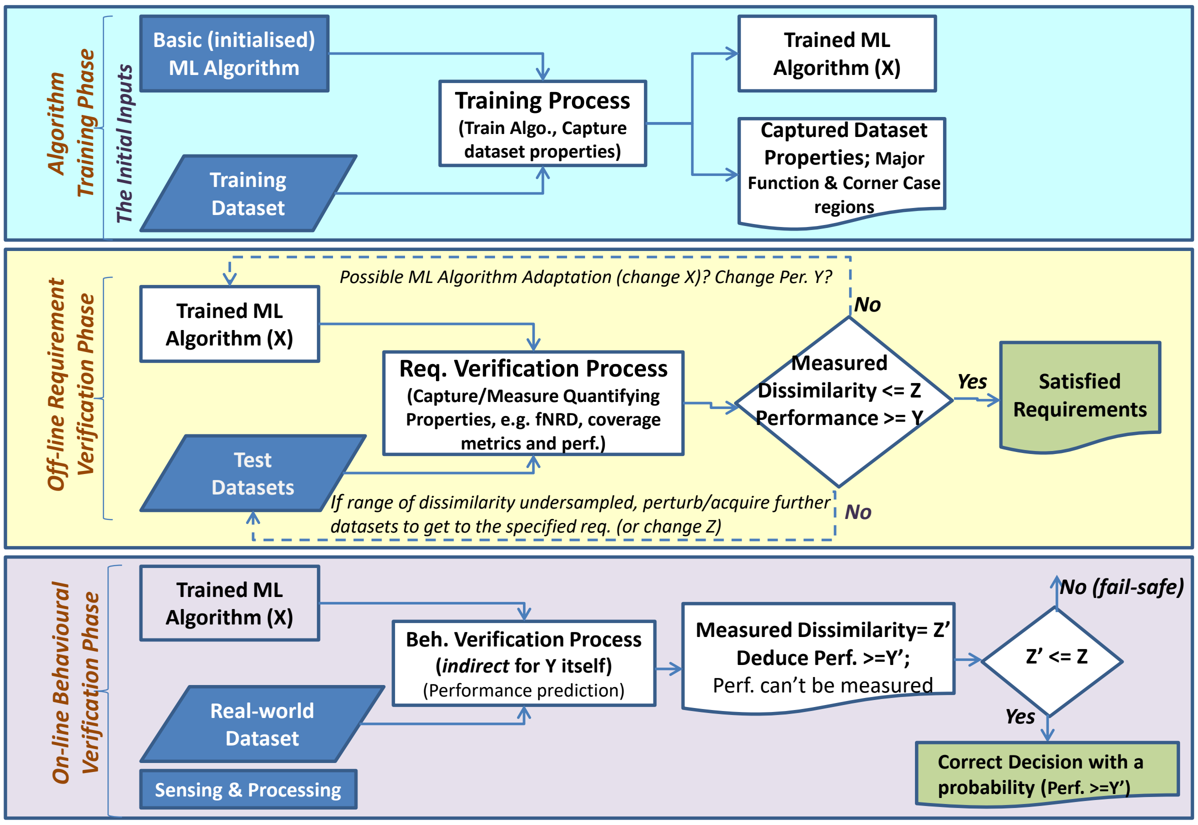
Feedback from the deployed system in the vehicle records anomalous events, inputs & behaviours. Those inputs and behaviours outside of the constraints of the simulation need to be assessed – possibly by widening the inputs to the simulation to cover these newfound scenarios. With these updated simulations the system can assess if these previously unexpected inputs are suitably handled, and we can probably extend the input data and therefore the constraints to reflect this newfound information. Or the simulation identifies the system does not behave as desired and some other course of action is required.

Design and Operation

The flow outlined above focuses on the operational flows; responding to events in the live system, assessing them, and making updates as necessary. This process is preceded by a design phase that results in a sufficient simulation model and body of evidence to commence deployment into a real system.

For a machine learning based solution the training and initial requirement verification phases occur 'off-line' before deployment of the system.

- **Dynamic Verification - Technical Requirement Spec.:**
  - **A Classifier X should perform at >=Y for dataset dissimilarity <=Z including generalisation capability.**



Algorithm Training Phase
The Initial Inputs
Basic (initialised) ML Algorithm
Training Dataset
Training Process
(Train Algo., Capture dataset properties)
Trained ML Algorithm (X)
Captured Dataset Properties; Major Function & Corner Case regions
Off-line Requirement Verification Phase
Possible ML Algorithm Adaptation (change X)? Change Per. Y?
No
Trained ML Algorithm (X)
Req. Verification Process
(Capture/Measure Quantifying Properties, e.g. fNRD, coverage metrics and perf.)
Test Datasets
Measured Dissimilarity <= Z Performance >= Y
Yes
Satisfied Requirements
If range of dissimilarity undersampled, perturb/acquire further datasets to get to the specified req. (or change Z)
No
On-line Behavioural Verification Phase
Trained ML Algorithm (X)
Beh. Verification Process
(indirect for Y itself)
(Performance prediction)
Real-world Dataset
Sensing & Processing
Measured Dissimilarity= Z' Deduce Perf. >=Y'; Perf. can't be measured
Z' <= Z
No (fail-safe)
Yes
Correct Decision with a probability (Perf. >=Y')

The suitability of the solution needs to be measured and only accepted for deployment once the necessary levels have been achieved.

The TS transaction is used to record simulation results. This could be used to record the results from the 'Off-line Requirements Verification Phase' above.

Where do limits for acceptability come from, are they represented in a TSR transaction, in the assumptions / constraints? Is this only intended to cover an update from an on-line system, or does the designer create the initial TSRs to create the suitable system.

System boundaries

For the purposes of this analysis, a system boundary needs to be defined to separate elements which are internal to a system and elements which are external to a system. A CAV system will be used as an example (Figure 1). In Figure 1, the CAV can be seen to be containing AI, HMI, Drive Control, Comms and Sensors, i.e. the components and sub-systems present on the CAV and required for the CAV to operate successfully. All of these except Comms are considered to be internal to the system. Comms possesses internal and external elements, for sending data inside and outside the CAV respectively. Sensors are considered to be internal to the CAV, as they are part of the CAV, and are not external to it. All the other systems shown in Figure 1 are considered to be external to the CAV. Changes to external systems should have only a minimal effect on the purpose of the CAV internal elements.

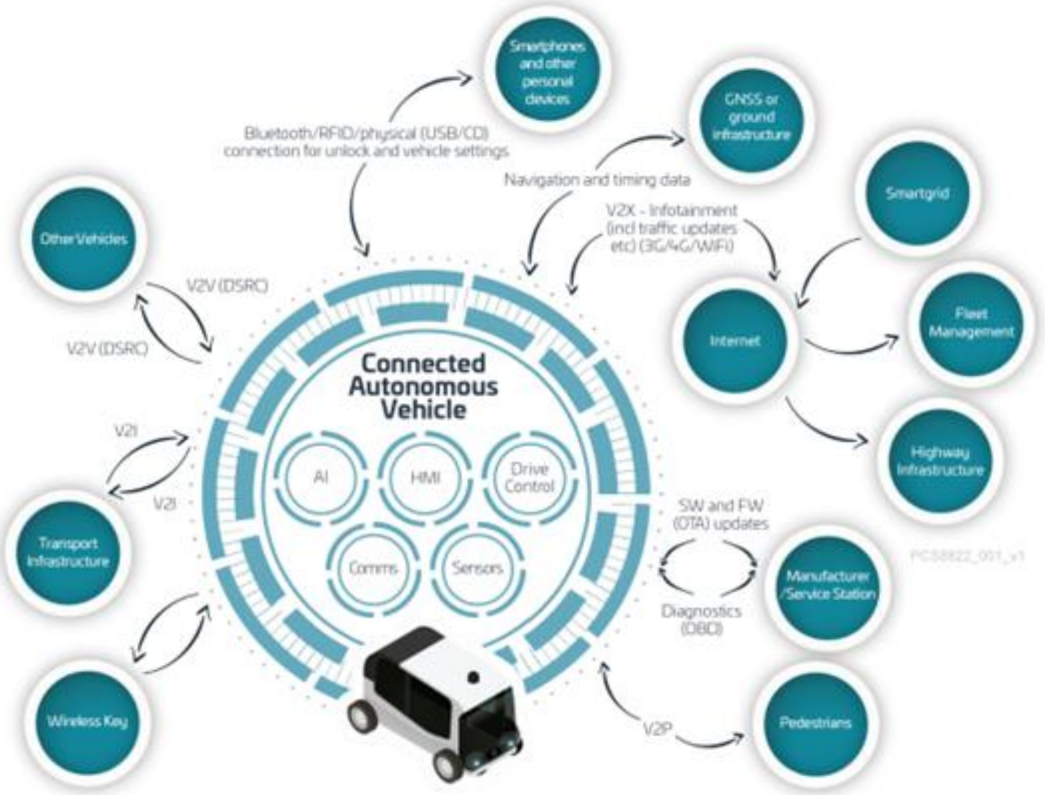

Smartphones and other personal devices
GNSS or ground infrastructure
Bluetooth/RFID/physical (USB/CD) connection for unlock and vehicle settings
Navigation and timing data
V2X - Infotainment (incl traffic updates etc) (3G/4G/WiFi)
Smartgrid
Other Vehicles
V2V (DSRC)
V2V (DSRC)
Internet
Fleet Management
Connected Autonomous Vehicle
AI
HMI
Drive Control
Comms
Sensors
V2I
V2I
Highway Infrastructure
Transport Infrastructure
SW and FW (OTA) updates
Diagnostics (OBD)
Manufacturer /Service Station
Wireless Key
V2P
Pedestrians

Figure 1: Connected Autonomous Vehicle internal and external systems.

System balance

Any system possesses resources, data and processing need which need to be designed to be in the right balance in order for that system to work effectively and efficiently (Figure 2).

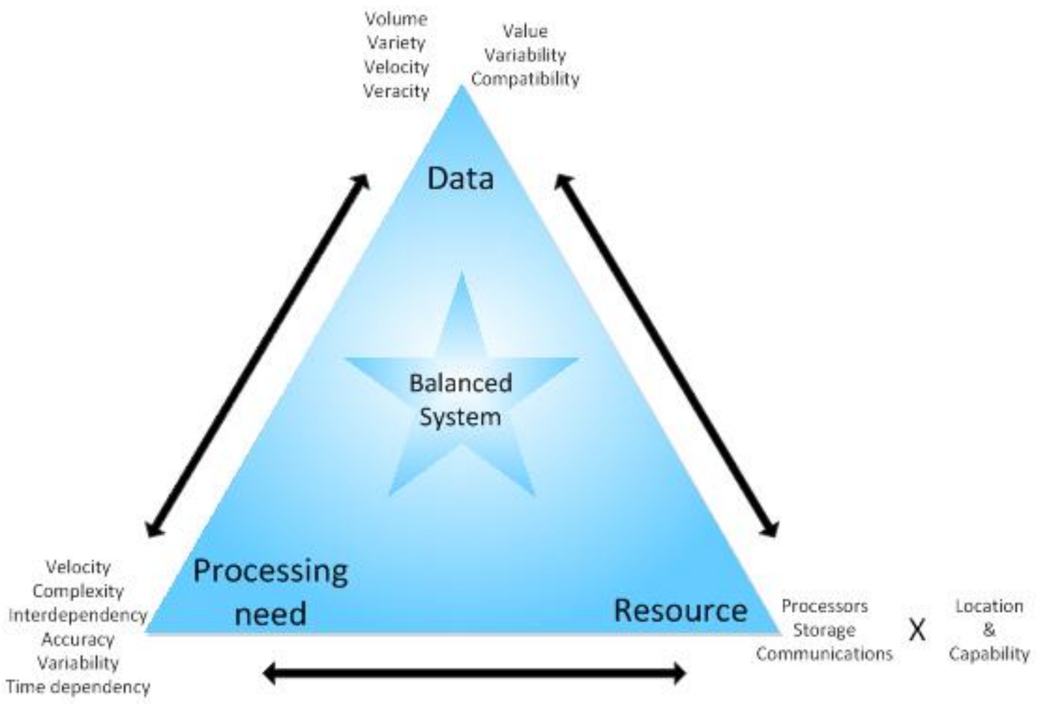


Figure 2: Scalability pyramid

Data includes any data and information input into the system, present within the system and output by the system. Data aspects include 7 of the V's of Big Data – Volume, Variety, Velocity, Veracity, Value and Variability[1], plus, the issue of compatibility.

- Volume – the quantity of the stored and generated data, and the size of an item of data.
- Variety – the types and natures of the different data items.
- Velocity – the speed at which data is generated, processed and made available to the system.
- Veracity – the truthfulness or reliability of the data, including the data quality.
- Value – the worth of the data, why is it useful, what can it do for the system?
- Variability – the changing nature of data, including formats, structure or source.
- Volatility – how long before an item of data is irrelevant, historic or no longer useful?
- Communication / bandwidth – what amount of data needs to be communicated?

Resources include data processors, data storage and data communication. For each of these, the quantity available, the capabilities of each (processing volume and processing speed), and the location of each need to be considered.

Processing needs focus on how to transform data inputs to produce data outputs. Processing aspects include velocity, complexity, interdependency, accuracy, variability and time dependency.

- Velocity – how quickly are the outputs required?
- Complexity – how complex is the processing required to meet the processing needs?
- Interdependency – what processes depend on the output of other processes?
- Accuracy – how accurate do the outputs need to be?
- Variety – what different forms of processing are required to meet the processing needs
- Time dependency – how do the processing needs vary with time, e.g. peak time versus off-peak time?

Effects of increasing the size of a System of Systems.

When a system is integrated into a system of systems, or as the number of systems in a CAV SoS increases, the purpose of the systems internal to each CAV are only minimally affected, but it is now exposed to external inputs and requirements. The internal processes are still required to operate the CAV in a safe and secure manner. The major effects on the SoS are increases in the amount of data now present and the number of interactions between the SoS components. Interactions can be purely informative i.e. exchange of data, or may require systems to decide upon and perform some action. The system's data, processing and resource capabilities and activities need to change in relation to each other to maintain a balanced system. Increased volume and variety of data leads to increased processing need and an increased resource requirement.

Considering the three aspects of the scalability pyramid, the following effects and issues are noted:

- Data
    - The volume of data present in the SoS increases as each additional system adds its own data to that available.
    - What data and when does a CAV need it for successful operations? Will this be a data push or data pull mechanism?
    - Only some of a system's data has value to other systems, so not all of it needs to be shared.
    - The correct types of data need to be stored for future evidence production.

    - **Conclusion** - Data processing, storage and communication requirements will increase as a result of the need to manage the increased data volume (amount) and data size (bandwidth requirements).

- Processing
    - Additional tasks introduced by increasing SoS complexity:
        - Coordination and management of the SoS systems – controlling how the systems interact to achieve SoS operational success.
        - Management of data communication between systems - determining what, when and who to share a data item with, and checking the received message accuracy / data loss. Communications management processing will include monitoring and predicting the available communications bandwidth and traffic, then balancing this against the importance of the data, in order to find the optimal time to send data.
        - Data assessment – determining what data to send or store. For example, only transmitting the salient points rather than all the raw data during busy times or when communications bandwidth is limited
        - Data summarisation – reducing the size of data items to send or store, from large amounts of low meaning data into smaller amounts of higher meaning information. Note this will take into account the requirements of digital forensics.
    - It is expected that there will be limited processing ability present on some systems, due to the physical size of the systems and of the processors. For example, a CAV will have limited processing capability. Each CAV's processors will prioritise essential CAV operational processing, but this will require processing management processes.
    - **Conclusion** - Extra processing will be required, for applying existing processes to larger volumes of data and for managing the SoS.
- Resources
    - Storage
        - Data storage has cost associated with it. Options include:
            - Everything – observations and all processing outputs.
            - Observations only – this assumes all processing outputs can be generated efficiently on request, and may require repeated processing to produce the same processing output when required.
            - Observations and frequently used outputs – infrequently used outputs are generated on request.
            - Observations and summaries.
            - Data required for evidence production and forensic analysis?
        - For a SoS, the location of data storage is also a factor. Systems such as CAVs have limited storage capability, thus only data essential to a CAV's operation and recently acquired data will be stored on-board. Once data has been successfully transmitted elsewhere, non-essential data can be over-written. The bulk of the data will be stored off-line and efficient data access required.
        - **Conclusion** - determining what to store and where to store data affects the economic feasibility.

    - Communications
        - Some communication connections will be transient, e.g. during a CAV's journey. The length of time a CAV is in contact with other systems that it passes will vary, based on the travel velocity, the types of communication media involved and the range capability of the communication media.
        - Networks of connections can be formed to extend the range over which data can be communicated. However, during a journey, it may be difficult to determine if a message has been delivered to its intended recipient due to the dynamic nature of such a network. Issues of data security also need to be considered as the network may be made up of many different types of system. Data encryption will increase the processing load.
        - Communications bandwidth and load must be balanced against message volume (amount and size). Large messages can be sent during times of low traffic or when bandwidth is higher. Alternately, only the salient points can be transmitted instead of large amounts of raw data.
        - **Conclusion** – Appropriate communications need to be available within the SoS, alongside the additional processing required to manage the increased communications.
    - Processors
        - Location - Some activities can be performed elsewhere, i.e. off-CAV at some central repository.
        - Capability – Additional processing is needed as a SoS grows. The nature of what processors are required will be influenced by design decisions regarding how best to perform the processing (see below).
        - **Conclusion** – resource location and capability are needed to support the increased data and its associated SoS requirements.
- Other issues:
    - Good compatibility between systems within a SoS will reduce costs. This will require the development and agreement of interfaces and standards. Without these, extra cost will be incurred to develop the bespoke software required for system interaction.
    - Timing of activity – This includes data processing and data communication. It will prioritise essential CAV operational processing. Other processing may have to wait. Some activities can be done when things are quiet or when CAVs are in certain states, e.g. recharging.
    - Security requirements will increase as the amount and variety of data and systems within a SoS increase.
    - Design decisions - Technology exists to facilitate scalability for large and growing systems, e.g. Software as a Service (SaaS) which allows users to connect to and use cloud-based apps over the internet, Cloud storage and processing, micro-services, Amazon Web Services (AWS) and SW languages e.g. Go.

Scalability solutions

From the discussion of the effects of scalability above, the key part for making scalability work is getting efficient and appropriate processing, storage and communication of data. The main things for this are increasing the resources and having appropriate software for managing them.

Initially as the number of systems goes up, the amount of data goes up. This forces up the resource and processing needs, which in turn increase the amount of software (processing) required. Software requires resources, which create processing need and the cycle continues until a balance point is reached (Figure 3). At this point, the amount of resource and software balance each other's needs in order to support the data need.

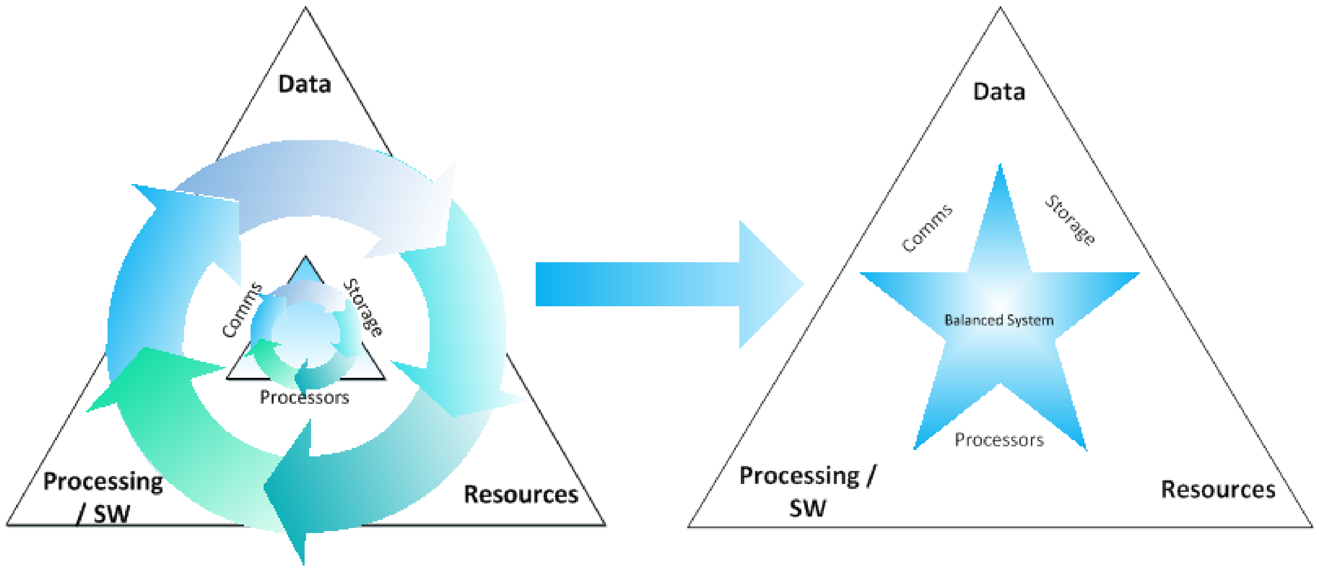


Figure 3: Moving from increasing requirements cycle to a balanced system.

Achieving economic feasibility

Good design is the key to achieving economic feasibility. By considering the requirements of a growing SoS at design time, it is possible to develop a system which uses the best and most appropriate technology in terms of hardware and software. Additionally, it is easier and cheaper to scale a system which has been designed for scalability, than to retrofit scalability to an existing system. As mentioned earlier, the use of common, agreed interfaces and standards between components or systems within the SoS, also contributes to keeping the costs down. While both of these have an initial financial outlay, they reduce the later and ongoing costs.

Blockchain transactions

We identify four types of transactions, namely Configuration (TC), Event (TE), Simulation Request (TSR) and Simulation (TS). Each transaction type contains two fields, subsystem and use_case, making them enough generic to be applicable to different aspects of the vehicle.

Furthermore, each transaction type is defined as follows:

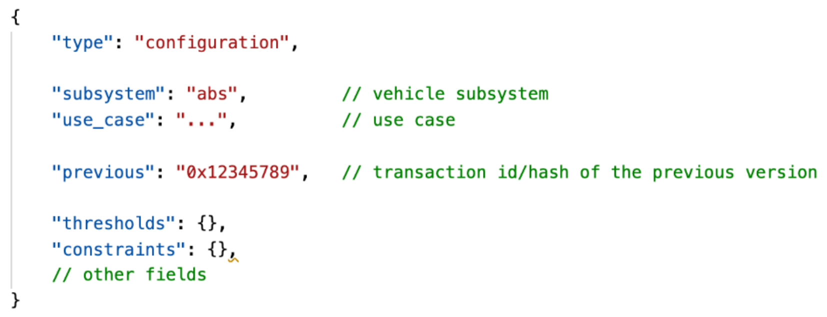
- **Configuration (TC)**

  Inserted by: Monitor/Adjust block

  Used by: All blocks

  The configuration transaction defines a specific and immutable set of constraints, thresholds and parameters. Transactions fields depend on the {subsystem, use_case} pair; in our example, the ABS stability analysis requires a set of thresholds and constraints.

  The field “previous” maintains a link to the previous version of the configuration, before adaptation.



```
{
    "type": "configuration",

    "subsystem": "abs",         // vehicle subsystem
    "use_case": "...",          // use case

    "previous": "0x12345789",   // transaction id/hash of the previous version

    "thresholds": {},
    "constraints": {},
    // other fields
}
```

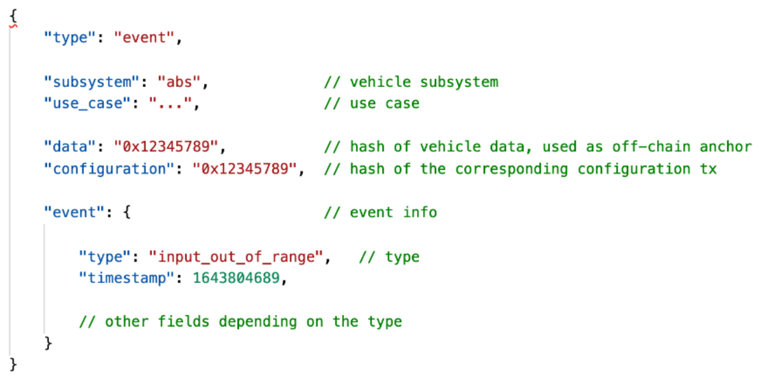
- **Event (TE)**

  Inserted by: Vehicle / Environment

  Used by: Monitor/Adjust block

  The event transaction contains all the information related to an anomaly reported by the vehicle. It is connected to:

  - Vehicle data, using a hash-based file anchor (for example IPFS storage)
  - Configuration, using the hash of the corresponding TC



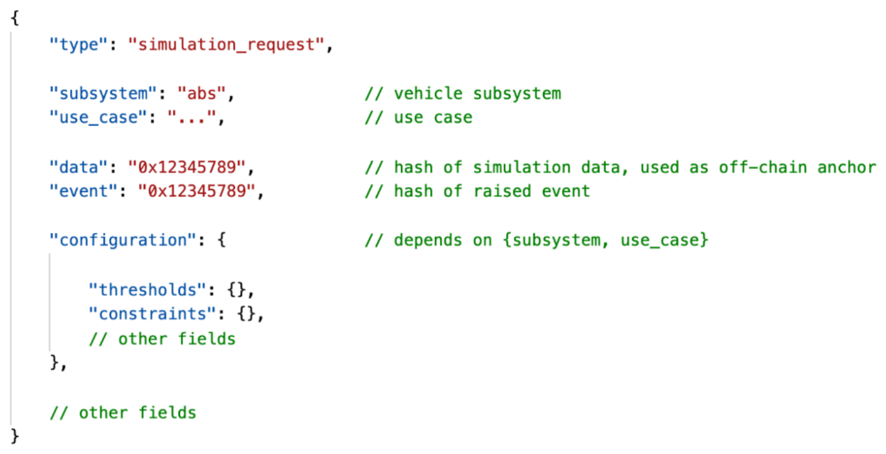
```
{
    "type": "event",

    "subsystem": "abs",              // vehicle subsystem
    "use_case": "...",               // use case

    "data": "0x12345789",            // hash of vehicle data, used as off-chain anchor
    "configuration": "0x12345789",   // hash of the corresponding configuration tx

    "event": {                       // event info

        "type": "input_out_of_range",    // type
        "timestamp": 1643804689,

        // other fields depending on the type
    }
}
```

- **Simulation Request (TSR)**

  Inserted by: Monitor/Adjust block

  Updated by: Simulation block

  This transaction is written by the Monitor/Adjust block to request a new simulation, arose from a specified event.



```
{
    "type": "simulation_request",

    "subsystem": "abs",              // vehicle subsystem
    "use_case": "...",               // use case

    "data": "0x12345789",            // hash of simulation data, used as off-chain anchor
    "event": "0x12345789",           // hash of raised event

    "configuration": {               // depends on {subsystem, use_case}

        "thresholds": {},
        "constraints": {},
        // other fields
    },

    // other fields
}
```

- **Simulation (TS)**

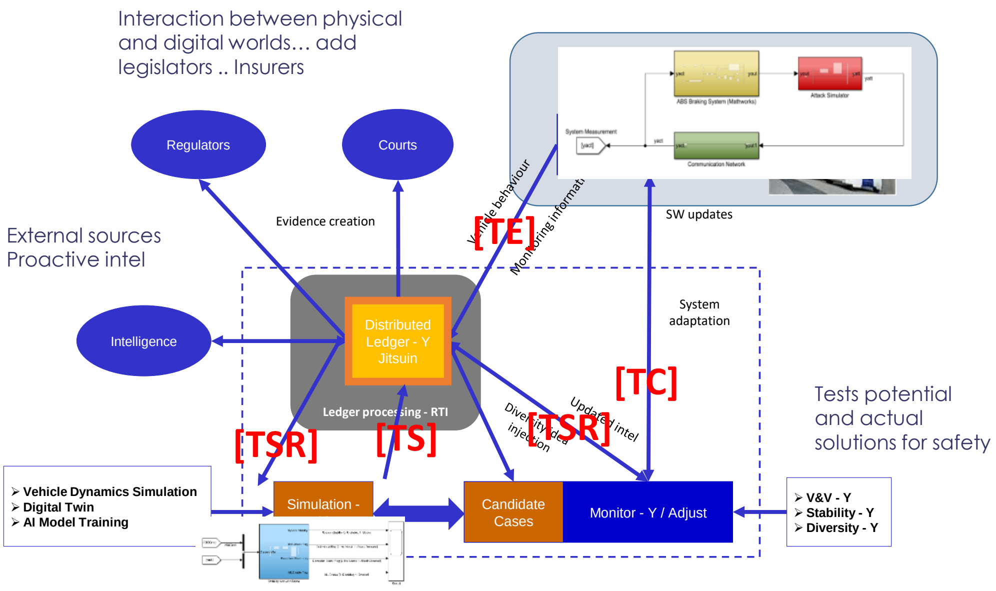
  Inserted by: Simulation block

  Used by: All blocks

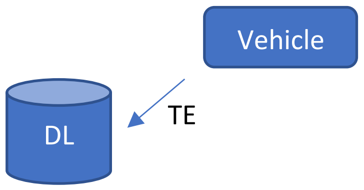
  This transaction contains the outcome of a simulation.

```
{
    "type": "simulation",

    "subsystem": "abs",             // vehicle subsystem
    "use_case": "...",              // use case

    "data": "0x12345789",           // hash of simulation data, used as off-chain anchor
    "event": "0x12345789",          // hash of raised event

    "configuration": {              // depends on {subsystem, use_case}

        "thresholds": {},
        "constraints": {},
        // other fields
    },

    "outcome": []                   // simulation outcome

    // other fields
}
```

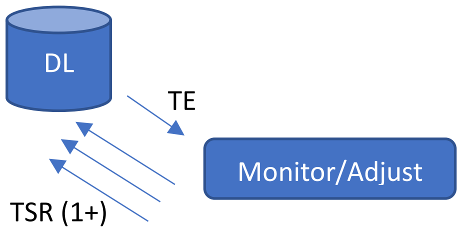
Transaction flows



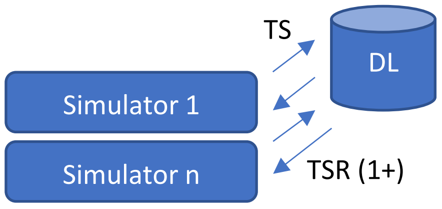
Using a smart contract, the vehicle stores a new TE transaction



Using a smart contract, the Monitor/Adjust block retrieves the newly added TE transactions and, starting from the current configuration linked in the event, it prepares one or multiple new plausible configurations for simulation. Such simulation proposals are stored as TSR transactions



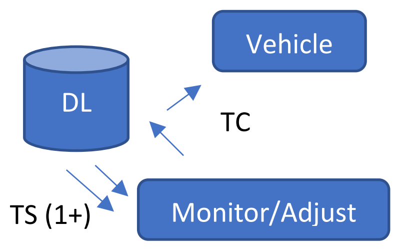
Using a smart contract, each simulation block retrieves the newly added TSR transactions and verifies if they can be satisfied (same {subsystem, use_case}). If so, the simulation is run and the outcomes will be stored in a new TS transaction.



Simulation requests (TSR) and results (TS) contain model, use_case, configuration, test data and all the other info needed to provide a full picture for future assessment and validation.

The data-link in TE refers to vehicle data, whilst data-link in TS refers to test-data. Vehicle and test data are stored off-chain.

Using a smart contract, the Monitor/Adjust block collects all the simulations (TS) and, considering the outcome, it might decide to promote a new configuration, creating a TC transaction. The latter is stored into the DL and possibly forwarded to the vehicles, thus used in identifying new events.



Breaking the Brakes

Breaking the Brakes as an Exemplar

Shmyglya[1] investigated the attack surface of modern cars with the aim to find areas of weakness as the industry is moving towards connected autonomous vehicles and mobility services. In comparison to traditional cars that were designed as self-contained systems, the modern car offers a variety of I/O ports and external connection points to enable features such as vehicle to vehicle communication, satellite navigation or advanced driver assistance, and to accommodate on-board entertainment as well as telematics. This creates opportunities for different types of attack, exploiting vulnerabilities across the attack surface from attacks that require direct access to the in-vehicle communication network, via attacks that require close co-location without direct access (e.g. using Bluetooth) to fully external attacks that can be launched without direct access or co-location at the time of attack, but may require direct access initially for setup.

The CAN bus has been identified as a point of particular vulnerability. It provides the communication network and protocol widely used in modern cars, where a variety of dedicated Electronic Control Units (ECUs) from different domains can exchange information without a central host. When it was first designed, in the late 1980s, the CAN bus was intended to be used for closed systems. Thus, cybersecurity was not a design objective at the time. However, since then connectivity has increased significantly. The individual CAN-bus based networks now connect to a central gateway, which can be accessed by the On-Board Diagnostics (OBD) port as well as the telematics and infotainment domain. This invalidates the original design assumptions and results in an inherently insecure communication infrastructure upon which safety-critical functionality is built. For example, steering and braking ECUs are connected to a designated high-speed CAN bus for safety-critical components that operate under real-time constraints.

Communication on the CAN bus relies on multi-master message broadcasting, i.e. any node can send messages and these can be received by all nodes connected to the network. The frame structure does not require source nor destination fields. Furthermore, there is no formal authentication on the CAN bus, which naively assumes that all nodes communicating on the network are legitimate and trustworthy. Clearly, this need not be the case. In fact, this can be exploited by an attacker who could inject malicious frames into the network in various ways. To exploit the arbitration, for instance, when a continuously transmitted stream of malicious frames is given highest priority then legitimate frames from safety-critical nodes get blocked. This is termed a Denial-of-Service (DoS) attack. Alternatively, brute-force flooding of the network can consume bandwidth and computational resources, leading to timing delays. The result is a violation of the real-time constraints required for the system to meet safety-critical requirements, thereby potentially endangering the users. In addition, carefully crafted malicious frames could confuse the control systems, e.g. selectively or randomly triggering the ABS without user input could lead to very dangerous situations.

How easy it is to launch such a DoS attack has been explored in Shmyglya[19] using a prototype test board as shown inFigure 14. The board includes an Ethernet network switch to which an input controller ECU and a motor controller ECU have been connected using Ethernet connections. The motor controller ECU is also connected to a motor. In practice, the input controller receives physical input from the user, e.g. braking. This is then processed, and the motor controller stops the motor as required. Communication on the test board has been set up to mimic the CAN-bus infrastructure and communication protocol as much as possible.

[1] Shmyglya, A. (2020). Breaking the Brakes: Spoofing and Denial of Service Attacks for Safety Critical Vehicle Components.

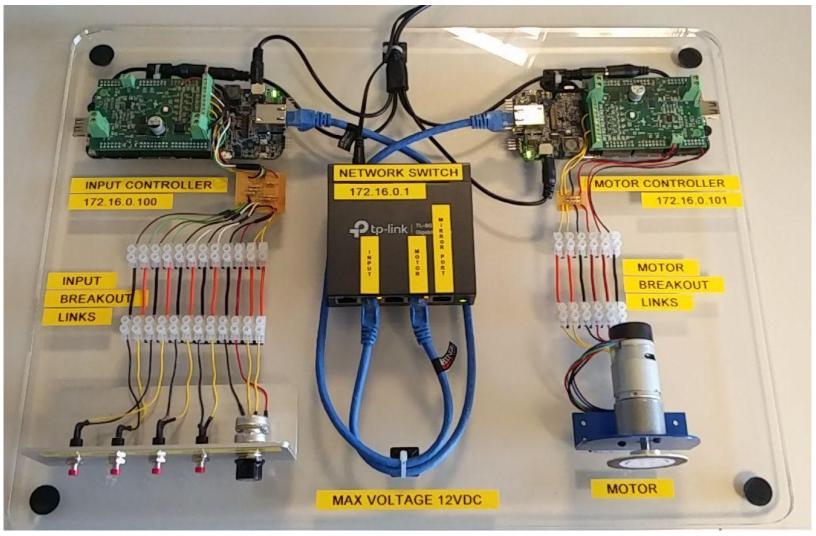
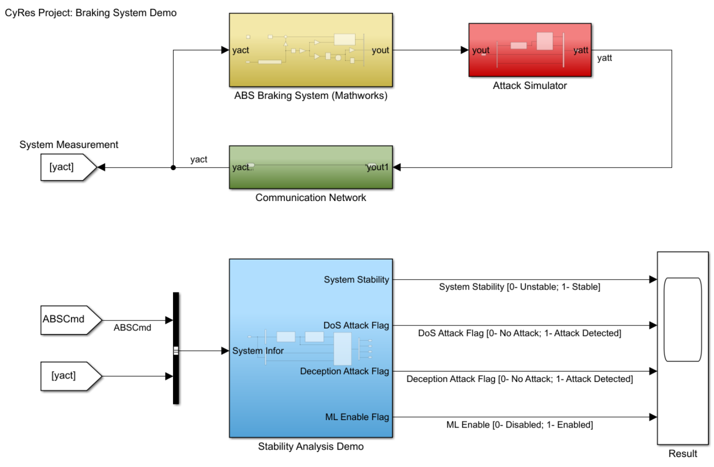


Figure 4: LAN with input controller (top left), motor controller (top right), network switch (central) and motor (bottom right).

Experimental evaluation revealed that a DoS attack could successfully be launched based on a UDP packet flood for a broadcast destination IP. In some cases, the injection of as few as 50 malicious packets in quick succession would break the application layer of the input controller, later the motor controller would also stop sending requests. A full reset was required before normal operation could be resumed. This illustrates the systematic insecurities present in networks of this type and calls for significant improvements to ensure the safety and security of vehicles.

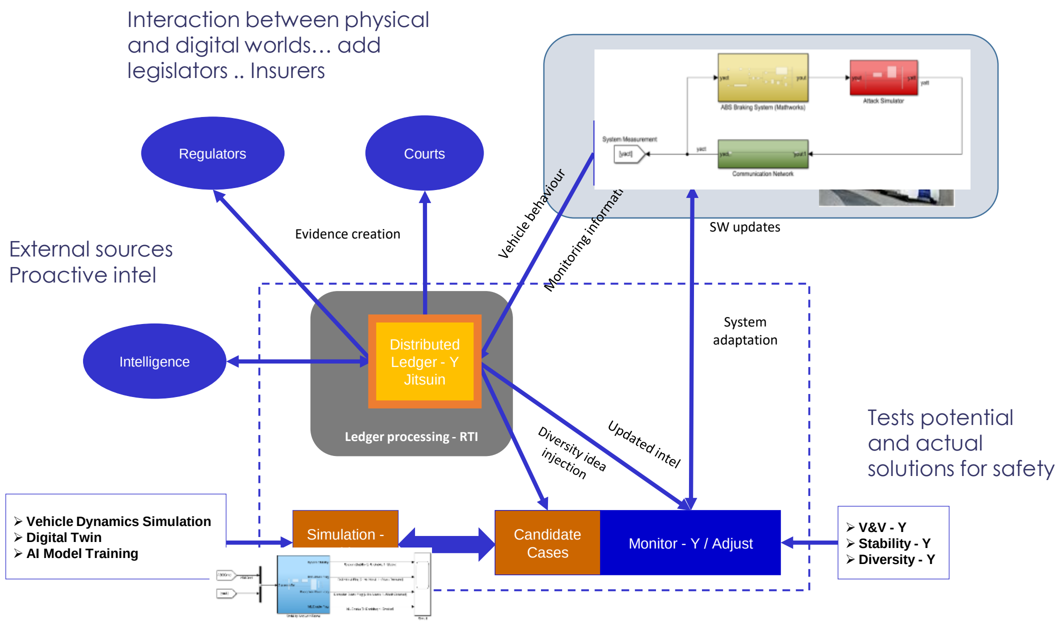
The ABS Braking System in ResiCAV+



CyRes Project: Braking System Demo
yact
yout
ABS Braking System (Mathworks)
yout
yatt
Attack Simulator
yatt
System Measurement
[yact]
yact
yact
yout1
Communication Network
ABSCmd
ABSCmd
[yact]
System Infor
System Stability
DoS Attack Flag
Deception Attack Flag
ML Enable Flag
Stability Analysis Demo
System Stability [0- Unstable; 1- Stable]
DoS Attack Flag [0- No Attack; 1- Attack Detected]
Deception Attack Flag [0- No Attack; 1- Attack Detected]
ML Enable [0- Disabled; 1- Enabled]
Result

The top half is a representation of the real system, the variation in the performance of the communication network is an example of significant difference. The bottom half is the simulation resulting in the decisions of ‘suitability’ of the system.

The representation of the braking system is an instance of the system in the real system; the suitability analysis is an instance of a simulation. Below the elements are overlaid on the solution overview:



The storage and communication method to allow the multitude of systems and their simulators to communicate is the Distributed Ledger. The detailed implementation of a simulation, and the detailed test data used to drive a simulation and achieve a particular result needs to be identifiable and retrievable – generally configuration management encompasses these tasks. The actual implementation and actual test data does not necessarily have to be stored into the DL.

The real system should report events to the DL whenever inputs or outputs are outside of those assumed or achieved by the simulation. For example, if the delay in the communications network (green box) is outside of the range assumed in the simulation this needs to be recorded as an event.

The Monitor/Adjust processing has to take this event and determine how to update the simulation to reflect this. This updated assumption / constraint is an input to the simulator(s) and needs to be recorded with an updated result. Given updated results (either the simulations say the system still works OK, or it says it does not), decisions need to be made; e.g., requesting modifications to the system.

These events and integrations are essentially four interfaces:

Simulation input, assumptions and constraints
Simulation output and conclusions
Configuration of the real system
Anomalous event report from the real system

These four interfaces can be represented as four transaction types in the DL.

The goal would be to have several different instances of the system with different software implementations / configurations (this contains the significant difference if they behave / operate sufficiently differently) – i.e., multiple of the mustard/yellow boxes in the top right of the diagram. The multiple configurations would all be simulated – i.e., multiple of the blue boxes in the bottom left of the diagram. The configurations that ‘pass’ the simulation could then be rolled out as replacements of the configurations that ‘fail’ the simulation.

This suggests the core item of interest in the Distributed Ledger is the real system (the braking system), of which there are multiple versions / configurations made up of different software configurations.

Each of these versions / configurations has one or more related simulations that reflect their individual behaviour. For a single version / configuration there may be multiple simulations because there could be multiple ways to simulate the system that are all useful but independent. The validity of a simulation could also change over time – it may become known that a simulation is not giving the accuracy previously thought and so it might be replaced with an updated simulation.

For any simulation there are the set of constraints / assumptions (e.g., the range of network delays). Given these constraints / assumptions, a simulation implementation and the simulations test data it records its result (e.g., stable / unstable)

The real system records events in the DL when there is a violation of the constraints / assumptions used to simulate the particular version / configuration deployed to that instance of the system (i.e., the software build on a braking system). The monitor solution should trigger the re-assessment in the simulated environments.

The re-running of the simulations with the updated constraints / assumptions provides a set of data for decisions to be made, with these results in the DL they are available for post assessment and validation or review. The decisions made as a result of assessing these results (either by human or machine) also needs to recorded in the DL sufficiently to identify the results being relied upon and decision reasoning – e.g., versions / configurations 1, 2, 3, 5, 6, 7, 8 simulate as OK with the new constraints, version / configuration 4 does not – therefore change all instances using version/configuration 4 to one of the other version/configurations.

5. Some Conclusions and Further Work

This report gives a brief summary of the design concept of a stability analysis tool which can be used for resilient control development for CAVs. The developed tool is capable of:

- Working in real-time with low computational cost as it only consists of a set of checking conditions
- Working as a standalone tool that can be attached to the vehicle to monitor the vehicle stability
- Working within a fleet of CAVs with the use of blockchain technology to ensure the information transparency and reliability of the decision made by the tool

We then analysed the requirements and proposed a viable integration between the blockchain platform and Cyres system. In doing so, we identified four transaction types and defined their structure.

We also investigated the specific use-case of an ABS system, providing a set of sequence diagrams corresponding to the various phases.

We have presented a number of tools, techniques, methods and examples developed and investigated by the consortium and our wider academic partners including the Trustworthy Systems Lab in Bristol to illustrate specific aspects of the CyRes methodology.

In consultation with both academic and industrial sources of tools the 'Framework' demonstrated by ResiCav+ needs to be developed and refined so that it may act as the core for a developing ecosystem of tools from the UK and more widely. This should be complete to industrial quality by March 2023.

We have investigated and demonstrated how online verification of a complex cyber-physical system could be undertaken at scale and within reasonable time constraints.

It is important to understand that, for the proposed online simulation-based verification techniques to work, significant preparation in terms of instrumentation and implementation effort is required at design time. For instance, a suite of tests would have to be collected, specifically for the system to be monitored. This would include preparation of software agent parameters tuned and optimised for generalisability so that tests can be generated online if required. Also, algorithms to prioritise test cases based on operational feedback would be required. In addition, test cases would need to be labelled at design time for rapid test prioritisation, using e.g. different types of coverage data. A promising future area to explore in this context would be the role and utility of KPIs (Key Performance Indicators) and SPIs[2] (Safety Performance Indicators) which may prove useful for the prioritisation of tests concerning performance or safety metrics.

There is also an interesting research area surrounding the causal link between system or component failure and design changes. Understanding how changing system parameters may lead to failure at design time (using simulation) may better inform the identification of causal relations at runtime. Additionally, understanding the assumptions made for assurance cases allows formalising these in SPIs so that they can be monitored at runtime. Any violations of SPIs undermine the assurance case

[2] Koopman P., Wagner M. (2020) Positive Trust Balance for Self-driving Car Deployment. In: Casimiro A., Ortmeier F., Schoitsch E., Bitsch F., Ferreira P. (eds) Computer Safety, Reliability, and Security. SAFECOMP 2020 Workshops. SAFECOMP 2020. Lecture Notes in Computer Science, vol 12235. Springer, Cham. https://doi.org/10.1007/978-3-030-55583-2_26

and can be used as early indicators of system safety being compromised. For example, the fact that the tyre tread is insufficient for cornering without skidding may help distinguish between a cyber-attack on the braking system and a technical deficit that should have been picked up at the last MOT. Likewise, monitoring the available CAN bus bandwidth and flagging up when the bandwidth required to support safety-critical functions is not being achieved, can help identify DoS attacks at an early stage.

There was a general observation that the tools sought out and integrated for ResiCAV+ were useful because they were integrated into a framework and concern that critical parts of an engineering process might become inaccessible if they were tied up by any one party to the detriment of this and other industries. One suggested solution to this was to form a UK company limited by guarantee in which HMG has a golden share. This would then be responsible for licence techniques eg data science algorithms identified under this programme to those in the UK. The suggestion that by retaining residual UK right in the event of a sale the developing 'basket' of rights will ensure that we do not develop a method which we then can't use and by ensuring that it can be used in the UK will help to attract inward investment. The idea of a company limited by guarantee, if successful, should within the next 5 years form the basis of a sovereign IPR fund ensuring:

a. The freedom for UK cyber companies to operate in emerging markets from CNI, through health and mobility where safety and security are a consideration,
b. inward investment to the UK (in order to access that IP),
c. additional levers to prevent offshoring of high value skills and jobs
d. HMG value for money for research and other investments made.

With respect to the tools it was felt that a reference work defining a complete set of industrial quality tools necessary to operationalise the conclusions of this study within the ResiCAV+ framework should be developed and, once endorsed by the stakeholder community, be maintained. This reference was seen as very important and it was suggested that were an initial version of this reference available by December 2022 with an endorsed version available no later than Mar 2023 then this would be tremendously helpful in planning future investment and metricating how close we might be to having a full methodology supported by tools.

Again, with respect to tools it was observed that the cost to develop tools cannot, and should not, be borne by 1 party alone. A number of the contributors observed that were they to develop this for their own use then every other OEM and tier supplier would need to do the same. This would both result in high, unnecessary, and ongoing cost but would also run the significant risk of a 'wild west' in which key elements were tied up by one party to the detriment of others. Accordingly, a fund of not less than £150m over 3 years, with an additional £100m available over the subsequent 2 years, available to tool developers generating IPR in the UK should be set up by the UK Govt to support the development of a world leading tool ecosystem.

Tools developed or adapted for the 'Framework' and either publicly funded or using a significant contribution from public funding should be made available for licencing through the UK IPR Pool. The licencing terms must ensure that residual rights are retained in the UK in the event that the tool or IPR is sold.

With respect to skills, it was observed that the net outcome of UN regulation 155 with respect to cyber has been to increase the skills deficit to a point where surveys have concluded that there is generally a deficit in the order of 0.5m cyber engineers in the automotive engineering ecosystem. This is a deficit which at current rates would have a cost of £ 75,000,000,000 per annum were it to be satisfied; it was observed that the projected requirement is unachievable, and were it achievable unaffordable, and were it both achievable and affordable ineffective at scale. ResiCav+ has demonstrated that the requirement to reskill and the type of skills could be achievable, affordable, and effective by prioritising the industrialisation of cyber resilience instead of trying to grow a highly paid cottage industry. It was concluded that is important that this transition is achieved over the next 3 years before the cost of meeting regulations in the current manner makes this unaffordable.

ResiCav+ has demonstrated that it is feasible to scale a tool based cyber resilience methodology in a complex system with emergent properties such that it could operate at a per vehicle (and per subsystem) level. It has demonstrated that this tool-based method is not inconsistent with the current international regulatory framework and could be used by regulators in the next step of their forward planning. Critically it was observed that the inability to update in a timely manner vehicles where that would violate the ‘Type Approval’ would likely become a point of cyber-attack over the coming 5 years.

The dialogue with members of the automotive industry undertaken by ResiCav+ has suggested that in the context of CyRes and its concept of ‘significant difference’ one use of the ‘regulatory sandbox’ idea currently under consideration might be to allow updates to vehicles to take place in a timely manner where that might otherwise not be possible on the basis that the potential harm could be restricted. It was observed that this would be consistent with a complex vehicle being effectively at all times in a prototype phase.

The dialogue with members of the automotive industry undertaken by ResiCav+ has led to the observation that it is likely not possible to automate the feedback loop in all circumstances. Equally it has been observed that it is not possible to achieve the volume or rate of updates that are foreseen to be necessary by using a wholly manual process. It was concluded that the use of a distributed ledger recording the events and the decisions that were made, or were not made, on the basis of those events would be beneficial in ensuring that the operation of the engineering process in the face of a complex automotive system was defensible. It was noted that this was likely the type of evidence that would be necessary to meet the Law Commissions foreseen criteria for an autonomy provider.

Appendix A BlockChain Transaction Schemas for RESICAV+

Purpose

This appendix describes the guidelines for the integration of the private-permissioned Blockchain platform for use in Cyres systems including:

- the identification of transaction types
- the proposal of transaction payloads

Areas Out of Scope

This appendix does not consider the need of transactions' encryption and access control, whose evaluation and implementation necessarily stem from considerations, outside the scope of this section.

In addition, the details of how the data is stored off-chain is not defined since the specific implementation, for example through IPFS or Amazon S3, is completely transparent from a transaction perspective.

Requirements

1. Subsystems (or *models*) of the vehicle are sources of *events*, which are evaluated and simulated against candidate *use-cases*, to predict how the system will behave;
2. Models, use-cases and events are bound within a set of assumptions about the range of inputs, system constraints and simulation parameters, under the umbrella-term *configuration*;
3. Those events and behaviours outside of the constraints of the simulation need to be assessed by concurrent *simulations* with new parameters, as in the case of widening the inputs to cover these new found scenarios;
4. According to the outcome of these new simulations, the system might became capable of handling the previously unexpected inputs or events, eventually confirming the validity of the *newly generated configuration.* On the contrary, if the simulation identifies the system does not behave as desired, another course of action is required.
5. It should be possible to track the life-cycle of each configuration, specifically, the previous set of parameters and the simulation/event that caused its evolution.
6. In order to limit the inevitable growth of blockchain storage, and related costs, it is mandatory to store vehicle and simulation data off-chain and to maintain integrity using hashes as file-anchors.

Transactions

Four types of transactions have been identified so far, namely *Configuration (TC)*, *Event (TE)*, *Simulation Request (TSR)* and *Simulation (TS)*. Each transaction type contains two fields, `subsystem` and `useCase`, making them generic enough to be applicable to different aspects of the vehicle, this way satisfying requirement (1). Furthermore, there are some dynamic fields (`configuration`, `eventInfo`, `simulationInfo`) whose content depends on the specific transaction type, model and use case.

Format considerations

In all implementations of Distributed Ledgers, transaction payloads are simply a set of bytes without a specific format. It is responsibility of the application layer (and smart contracts) to agree upon a specific structure. The possible choices range from a binary representation, optimised for speed and size, to a human-readable format at the expense of more verbosity. In this study, we used a JSON format since it is natively supported in almost all programming languages, facilitating the development of dependent software components. On the other hand, JSON does not limit the injection of different fields, which is mandatory to achieve the generality of our solution. Other formats have been considered, for instance YAML, MessagePack, Google Protobuf, etc., each one with a different trade-off among speed, size and ductility, however, each alternative is still compatible with the considerations in the following sections.

Configuration Transaction (TC)

The Configuration Transactions satisfy requirements (2) and (5), defining a specific and immutable set of constraints, thresholds and parameters as follows:

```
{
    type: "configuration",
    subsystem: "abs",
    useCase: "test",
    previous: null,
    simulation: null,
    configuration: {
        DelayLim: 10,
        WindowSize: 10,
        dT: 1,
        dyLim: 5,
    },
}
```

- Field `previous` contains the hash value of the previous configuration, this way maintaining a backward link to previous versions. A value of `null` indicates this is the first configuration for corresponding model/use case.

- Field `simulation` contains the hash value of the simulation that validated the configuration. A value of `null` indicates this is a bootstrap configuration.

- Dynamic field `configuration` contains constraints, thresholds and parameters.

Event Transaction (TE)

The event transactions satisfy requirements (3) and (5), containing all the information related to an anomaly reported by the vehicle, defined as follows:

```
{
    type: "event",
    subsystem: "abs",
    useCase: "demo",
    data: 0x12345678,
    configuration: 0x12345678,
    timestamp: 164394689,
    eventInfo: {
        ...
    },
}
```

- Field `data` contains the hash value of the data associated to this event, stored off-chain (requirement 6).
- Field `configuration` contains the hash value of the associated configuration, used for the identification of the event.
- Field `timestamp` contains the timestamp of the even, expressed in Unix Epoch.
- Dynamic field `eventInfo` contains all the information regarding the specific event.

Simulation Request Transaction (TSR)

The Simulation Request transactions satisfy requirements (3) and (4), representing the request to start a new simulation in response to an event, defined as follows:

```
{
    type: "simulationRequest",
    subsystem: "abs",
    useCase: "demo",
    data: 0x12345678,
    event: 0x12345678,
    simulationRequestInfo: {
        configuration: {
        },
    },
}
```

- Field `data` contains the hash value of the data which will be used for the simulation , that could be different from the data behind the event (requirement 6).
- Field `event` contains the hash value of the originating event.
- Dynamic field `simulationRequestInfo` contains constraints, thresholds and parameters of the simulation.

Simulation Transaction (TS)

The Simulation transactions satisfy requirements (4), containing the outcome of a simulation, defined as follows:

```
{
    type: "simulation",
    subsystem: "abs",
    useCase: "demo",
    request: 0x12345678,
    simulationInfo: {
        outcome: {
        },
    },
}
```

- Field `request` contains the hash value of the corresponding simulation request.
- Dynamic field `simulationInfo` contains the outcome and other information regarding the simulation.

Configuration traceability

Figure 5 shows the relations among the different types of transactions. This emphasises and clarifies how it is possible to navigate backwards from a configuration to its generating steps, this way fully and immutably reconstructing its history.

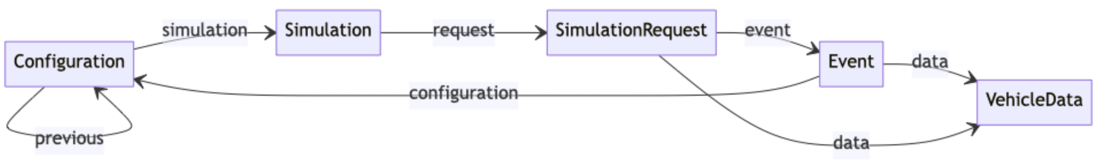


Figure 5: Relationship between transactions

Appendix B: A Tool For Blockchain Dynamic Update to Support CyRes

The ResiCAV+ team demonstrated a step forward in the methodology showing the use of simulation, the automatic generation of test cases and real time V&V for and on the simulator. The capabilities of Jitsuin's RKVST (pronounced 'Archivist') was used:

- to automate keeping track of the simulations, decisions, and updates (including ordering of these) with a view to being able to automate the production of compliance documentation and the inputting and outputting of threat information.
- As part of automating the compliant loading of certified baselines onto individual vehicles in a legally defensible way.
- Demonstrate the economic benefits of using Jitsuins technology in the context of making faster, confident decisions and reducing business risk

RKVST is a blockchain-powered platform that enables supply chain partners to safely share configuration, environmental, and operational data which results in traceable, accountable and trustworthy operations for connected industry. It records "When Who Did What to a Thing" to create a complete and tamper-poof life history of any cyber-physical asset.

This 'golden thread' of evidence can then be securely verified for proof of security and compliance by any authorized stakeholder: either the participants or their auditing authorities. The owner of the asset is in full control of which aspects of the life history they share, and with whom. If a part of the thread is shared then the external stakeholders know that the evidence is clear, complete, timestamped, and un-tampered. But parts of the thread that should remain secret, remain secret.

RKVST wraps a complex mix of blockchain, smart contracts, cryptographic key management and federated access control into a simple-to consume platform and API that is accessible and usable by regular IT teams in regular industries. It is brownfield-friendly (no endpoint agent or estate refresh required) and does not require federation of IT assets (every participant signs in with their own corporate sign-in and keeps their main data store private). This achieves:

- **High-Definition Control**: Ensure the right people have what they need to know and no more.
- **Assured Data Provenance**: Full traceability and lineage on all data sources that feed critical decisions.
- **Continuous Accountability**: Prove when who did what to any critical asset and build trust in digital operations.

In Figure 6: Blockchain registration and storage of activities as smartcontract transactions. Figure 6 below shows RKVST registering every critical activity in an Event record and storing evidence of that activity as smartcontract transactions on the blockchain. The Events are then collected together to create a complete Service History or 'golden thread of evidence' for configuration, operation, and handling (Figure 7). Figure 8 then shows the coordinating of players in the cyber physical security supply chain. Each participant plays their part in the 'team sport' of security and configuration management, and the other participants are immediately aware of changes and exceptions.

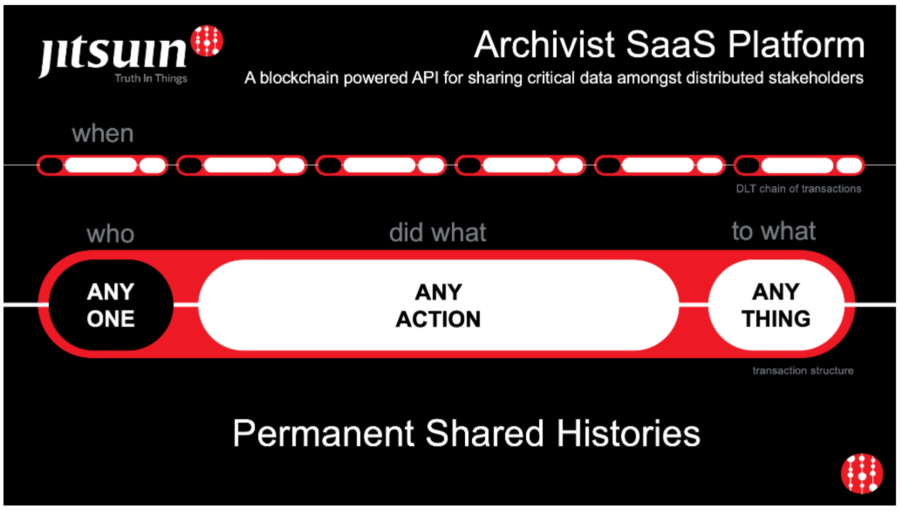
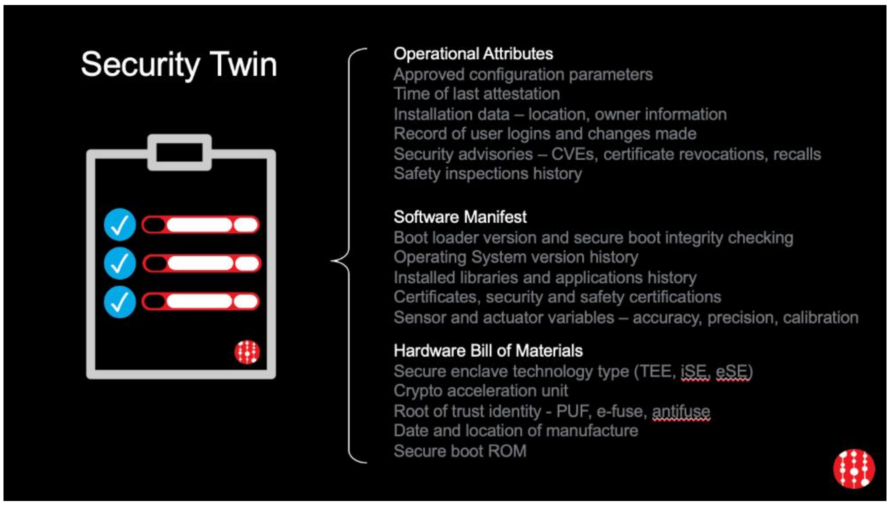

jitsuin
Truth In Things
Archivist SaaS Platform
A blockchain powered API for sharing critical data amongst distributed stakeholders
when
DLT chain of transactions
who
did what
to what
ANY ONE
ANY ACTION
ANY THING
transaction structure
Permanent Shared Histories

Figure 6: Blockchain registration and storage of activities as smartcontract transactions.



Security Twin

Operational Attributes
Approved configuration parameters
Time of last attestation
Installation data – location, owner information
Record of user logins and changes made
Security advisories – CVEs, certificate revocations, recalls
Safety inspections history

Software Manifest
Boot loader version and secure boot integrity checking
Operating System version history
Installed libraries and applications history
Certificates, security and safety certifications
Sensor and actuator variables – accuracy, precision, calibration

Hardware Bill of Materials
Secure enclave technology type (TEE, iSE, eSE)
Crypto acceleration unit
Root of trust identity - PUF, e-fuse, antifuse
Date and location of manufacture
Secure boot ROM

Figure 7: A 'golden thread of evidence' for configuration, operation, and handling

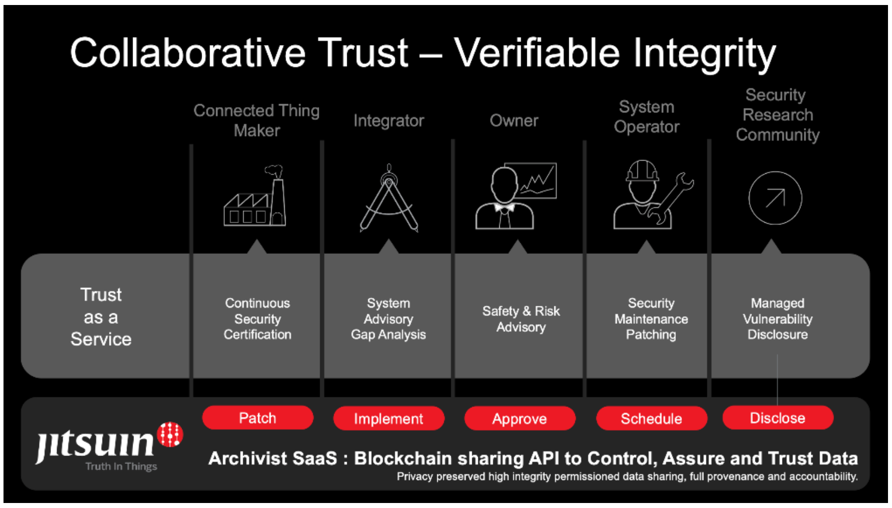

Collaborative Trust – Verifiable Integrity
Connected Thing Maker
Integrator
Owner
System Operator
Security Research Community
Trust as a Service
Continuous Security Certification
System Advisory Gap Analysis
Safety & Risk Advisory
Security Maintenance Patching
Managed Vulnerability Disclosure
Patch
Implement
Approve
Schedule
Disclose
jitsuin
Truth In Things
Archivist SaaS : Blockchain sharing API to Control, Assure and Trust Data
Privacy preserved high integrity permissioned data sharing, full provenance and accountability.

Figure 8: Coordinating players in the cyber physical security supply chain.

Market context

Digital Transformation today faces a problem: connecting together systems of systems requires data to flow across technical, organizational, and even geographic boundaries. In order to meet the promise of DX, components and systems need to trade operational data, act on each other's signals, and automate critical decision-making, but without unduly increasing risk exposure.

Herein lies a problem: for DX to work, data has to flow, but data security doctrine prefers to lock it up and keep it secret. Sharing data then becomes a complex exercise variously with federating identities, or making exception rules in protection systems, or configuring vast PKI systems.

Leading analysts now identify that problem:

"Secure and trusted data exchange across third-party ecosystems is a business necessity, however this often fails because of regulatory and trust concerns. Jitsuin RKVST offers businesses a scalable blockchain platform to enable trust and governance controls across sensitive information and asset sharing environments, and the confidence to advance new business models and outcomes."

Ralf Helkenberg, Research Manager, European Privacy and Data Security at IDC.

"Every digital business moment leads to a decision that is powered, or held hostage, by data and analytics, it is no wonder that chronic issues have become more acute. [because of] Siloed data, lack of trust, misalignment to outcomes, a focus on data for its own sake…"

Chris Howard, Chief of Research, Gartner

"In Digital Transformation with Connected Things […] 62% of projects fail because of security concerns"

Cap Gemini – Beecham Research - *Why IoT projects fail* - 2020

Confidenti ality
SECURITY
Integrity
Availability

Figure: The familiar 'CIA' triad

To paraphrase the Gartner opinion, confidentiality is a relatively solved problem, but availability and integrity are not. This is not a cryptography or tools problem so much as a trust and control problem: even if a digital signature is used by my supply chain partner, how do I trust that the data within has not been back-dated? Their IT department own the keys: perhaps they could cheat. Perhaps they could hide the data when I need it. Perhaps their security policies are not as strong as mine and they have allowed misuse of the keys.

RKVST aims to address these problems of collective trust by making every party accountable for their actions. If you do a good job then that is clear to see, but if something is missed or done incorrectly, that is equally readily found. And by modelling the data as Assets and Events, the platform avoids sharing 'data for its own sake' and concentrates on keeping a clean and actionable record of important operational data only.

Assets, Events, and Access Policies

RKVST is an asset-centric system. Users track real-world asset histories by first creating a tokenized record for the Asset, and then registering Events against that record as things happen to it. *The only way to change an asset property is by registering an event*, which means that rich provenance and lineage is automatically available for all properties of the asset. For example, if an IoT device is shown as having a particular firmware version, the system will also tell you who patched it, when they patched, why they thought that was the right thing to do and what evidence they used to come to that conclusion.

Access to the Asset record is mediated by Access Policies that enable fine-grained control of which Subject Principals (either a user in one's own organization, or a partner organization) can read and write the Asset, and which properties are available to them.

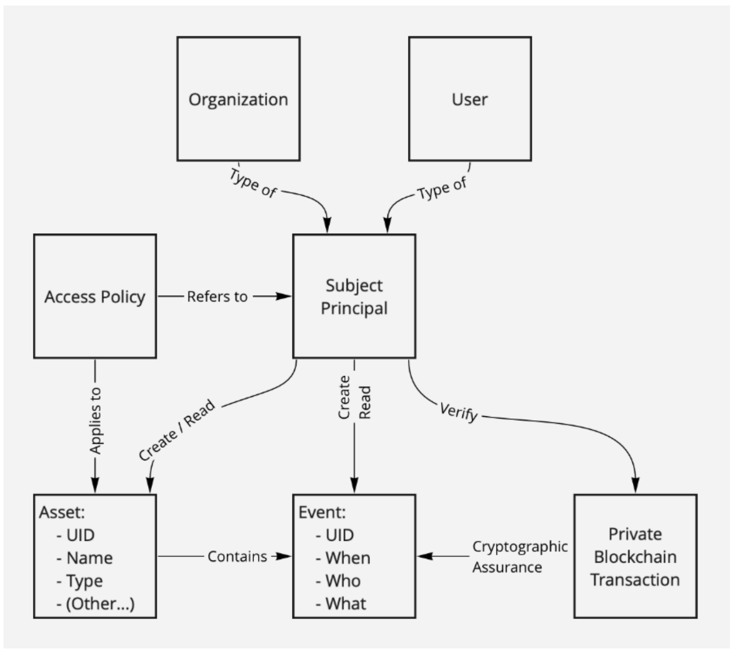


Figure 9: A simplified view of object relationships in RKVST.

Behind the scenes, RKVST converts access policies into cryptographic keying relationships that mean that data is only visible to the right participants, and can be stored on the blockchain and verified by all legitimate parties whilst not revealing secrets to others on the platform, *even when they have access to the underlying blockchain data*. All of this results in a golden thread of shared evidence of Asset interactions where authorized parties are certain that they see exactly the same version of data as all other authorized parties without the usual risks of version-mismatches, over-sharing, timestamp slip, or integrity issues. This can be seen in Figure 10 below showing how different organizations have appropriate visibility of Asset and Event data. Data they can see is verified by the blockchain, so that they know they see exactly the same view as the other participants. Data they are not allowed to see is never available under an encryption key they can access, ensuring privacy and secrecy where needed.

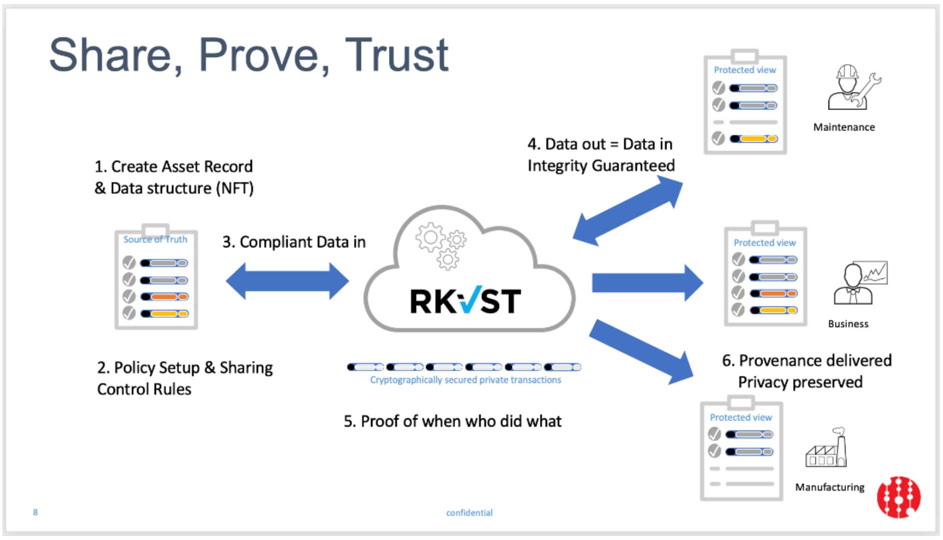

Share, Prove, Trust
1. Create Asset Record & Data structure (NFT)
Source of Truth
3. Compliant Data in
RKVST
4. Data out = Data in Integrity Guaranteed
Protected view
Maintenance
Business
2. Policy Setup & Sharing Control Rules
Cryptographically secured private transactions
5. Proof of when who did what
6. Provenance delivered Privacy preserved
Manufacturing
8
confidential

Figure 10: Appropriate visibility of Asset and Event data for different participants.

Witness statements and non-connected Assets

It is important to understand that Events in RKVST are a series of 'witness statements' made by authorized users: they are not (necessarily) sensor readings directly connected devices or dependent on any IoT platform attestation service. The owners of an asset can specify any authorized party to contribute life history Events to the golden thread.

This approach is powerful in cases such as connected infrastructure where each cyberphysical asset or system's record must contain a full and coherent mix of direct sensor readings from the device, automated messages arising from cloud systems, and manual data entered from inspections, maintenance, or other interactions.

In addition, real-world operations sometimes happen offline or in air-gapped facilities, and can only be reported later (for instance, when a maintenance crew complete their rounds and connect to a terminal or mobile device at the end of the day). For many practical reasons it is important to know when an event really happened: having them all listed as 17:30 PM and under the name of the manager or a system credential is not necessarily helpful. At the same time, for evidential value it is important to have proof that nothing could have been back-dated or modified.

For this reason, RKVST allows clients to set the "WHEN" and the "WHO" in an Event to reflect what happened in the physical world, and then adds a tamper-proof record of when the Event was actually entered into the system. Discrepancies between these values are easily discovered and transparent to all relevant stakeholders in real time, making fault finding and record fraud easily discoverable.

Timestamps

Once committed to the RKVST system, each lifecycle event record carries 3 separate timestamps:

- `timestamp_declared` - an optional user-supplied value that tells when an Event happened. This is useful for cases where the client system is off-line for a period but the user still wishes to record the accurate time and order of activities (eg inspection rounds in an air-gapped facility). If unspecified, the system sets timestamp_declared equal to timestamp_accepted (see below).
- `timestamp_accepted` - the time the event was actually received on the Jitsuin Archivist node's REST interface. Set by the web front-end system, cannot be changed by the client.

- `timestamp_committed` - the time the event was confirmed distributed to all DLT nodes in the value chain. Set by the DLT back-end system, cannot be changed by the client OR the Jitsuin web front-end.

Having these 3 fields enables users of Jitsuin Archivist to accurately reflect what is claimed, whilst also preventing tampering and backdating of entries.

User principals

Once committed to the RKVST system, each lifecycle event record carries 2 separate user identities:

- `principal_declared` - an optional user-supplied value that tells who performed an event. This is useful for cases where the user principal/ credential used to connect to the Archivist system does not accurately or usefully reflect the real-world agent (eg a multi-user application with device-based credentials).
- `principal_accepted` - the actual user principal information belonging to the credential used to access the Jitsuin Archivist node's REST interface. Set by the system and retrieved from the authorizing IDP, cannot be changed by the client.

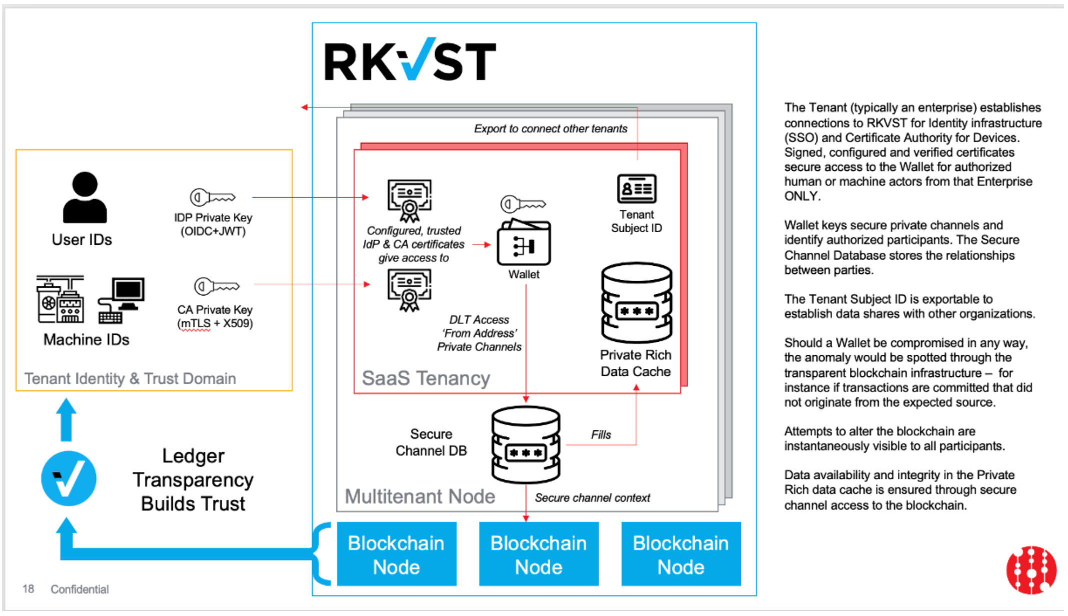
Tenancy model

While several deployment models are possible, offering varying degrees of flexibility and control, the main RKVST network runs as a multitenant SaaS on Microsoft Azure. Each participating organization (assumed to be a coherent legal entity) is assigned a 'tenancy' on the SaaS, with their own separate cryptographic keys, policies, and rich data store.

The tenant owner is completely in control of user access to their RKVST tenant: Jitsuin does not store user accounts at all. Instead, the tenancy is strongly bound to the customer's own Open ID Connect compatible identity provider (IDP) and Jitsuin verifies all accesses against that IDP. What this means in practice is that every company using the RVST is able to Bring Their Own Identity, adhering to their own IT policies, using their own trusted second factor technology, and having full control of revocation and user attribute management.

In this way, companies using Microsoft AAD identity and Microsoft Azure computing are able to collaborate seamlessly with partners who prefer Forgerock identity and Amazon compute. No federation is required. No new passwords on shared systems. No need for proprietary data to leave their old silos or be consolidated on one large central store. Organizations keep working the way they work, with the policies they already understand, and just enjoy the benefits of trustworthy data for the elements they choose to share.

When deciding which supply chain partners should be allowed to collaborate on which assets, tenants have the simple task of a one-time exchange of public identities, and then a flexible and powerful but familiar-looking read/write permissions interface. Jitsuin takes care of handling the blockchain specifics such as wallet keys, secure channels and so on. Figure 11 below sets out a simplified model of the domains of trust in the multi-tenant SaaS deployment of RKVST. Enterprise identity management (yellow) is handled by the tenant organization. Key management and blockchain interfaces (red) are handled by Jitsuin. The transparent blockchain record provides legally acceptable proof of when who did what.


RKVST
Export to connect other tenants
User IDs
IDP Private Key (OIDC+JWT)
Configured, trusted IdP & CA certificates give access to
Wallet
Tenant Subject ID
Machine IDs
CA Private Key (mTLS + X509)
DLT Access 'From Address' Private Channels
Private Rich Data Cache
Tenant Identity & Trust Domain
SaaS Tenancy
Secure Channel DB
Fills
Ledger Transparency Builds Trust
Multitenant Node
Secure channel context
Blockchain Node
Blockchain Node
Blockchain Node
18 Confidential

The Tenant (typically an enterprise) establishes connections to RKVST for Identity infrastructure (SSO) and Certificate Authority for Devices. Signed, configured and verified certificates secure access to the Wallet for authorized human or machine actors from that Enterprise ONLY.

Wallet keys secure private channels and identify authorized participants. The Secure Channel Database stores the relationships between parties.

The Tenant Subject ID is exportable to establish data shares with other organizations.

Should a Wallet be compromised in any way, the anomaly would be spotted through the transparent blockchain infrastructure – for instance if transactions are committed that did not originate from the expected source.

Attempts to alter the blockchain are instantaneously visible to all participants.

Data availability and integrity in the Private Rich data cache is ensured through secure channel access to the blockchain.

Figure 11: Simplified model of 'domains of trust' in a multi-tenant deployment of RKVST.

Decentralised and distributed evidence base

RKVST ensures that every tenant has fair access to their portion of the evidence base. If a dispute arises, or a supply chain partner goes bust, that party's data is not lost with them.

And while Jitsuin's SaaS represents a central point of access and configuration surface for each tenant, even Jitsuin is held accountable through each tenant's ability to verify their blockchain transactions and observe that the chain has not changed, and is the same as the one everyone else is using. Figure 12 below illustrate critical supply chain partners collaborating on the golden thread to create a trustworthy single source of truth.

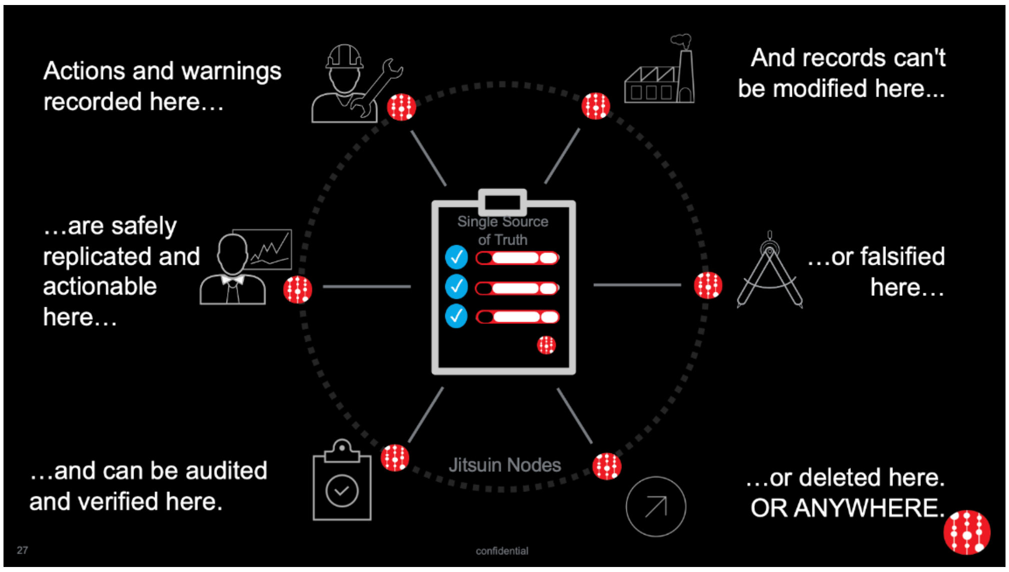

Actions and warnings recorded here…
And records can't be modified here…
…are safely replicated and actionable here…
Single Source of Truth
…or falsified here…
…and can be audited and verified here.
Jitsuin Nodes
…or deleted here. OR ANYWHERE.
27
confidential

Figure 12: Critical supply chain partners and a trustworthy single source of truth

Using RKVST Shared Asset Histories to find and prove problems

More efficient, provable audit processes

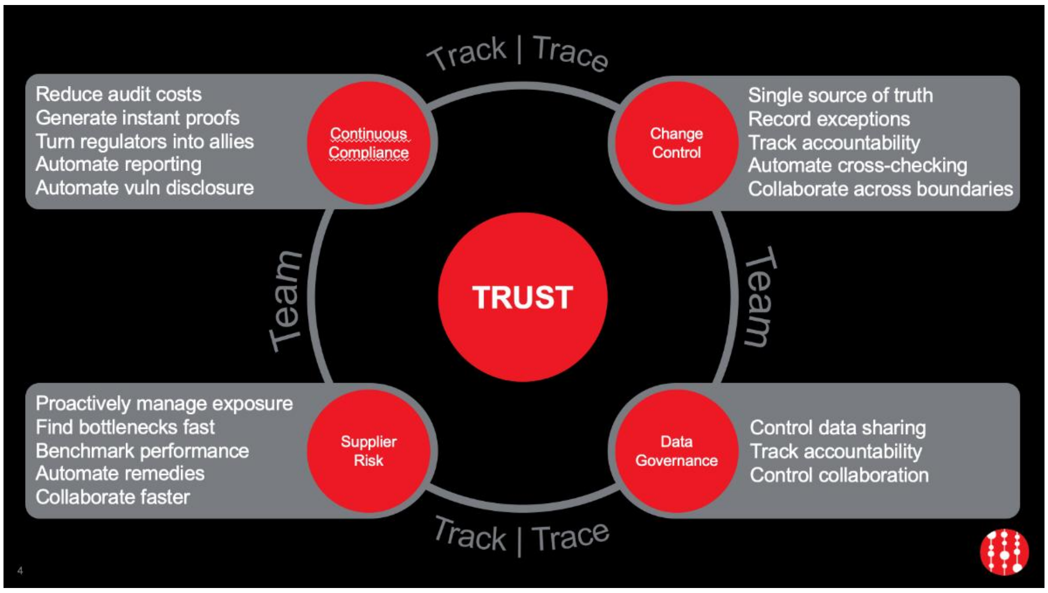


Figure 13: Addressing operational risk with trusted data, lineage, and asset provenance.

With all of this understood, Figure 13 sets out how operational risk problems can be addressed with trusted data, lineage, and provenance of the assets that created it. Using RKVST Shared Histories as a golden thread of evidence can clearly help to solve critical problems in cyber physical systems: issues not just of data and security but also of trust, risk, and liability. And because Event records can store evidence alongside them, it is simple to discover not only “what do we know now?” but also “what did they know at the time?”, “who authorised that?”, “were they acting within contemporary best practice?”.

There are four simple steps to this:

- **Record**: Every participant submits an honest record of WHEN WHO DID WHAT using their own IT systems and workflow
- **Consolidate**: The system arranges these records into a coherent Service History for each Asset, with reliable timestamps and integrity data
- **Analyse**: With the complete history available, hard questions that used to span organizational boundaries and data stores can now be asked easily: is this device up to date? Where did this configuration come from? Did anyone touch the asset while it was in an alarm condition?
- **Decide**: Given all you know about the history and handling of this asset, how much do you trust it right now? Is it in a fit state to have the interaction you want? Are you confident you have the regulatory or other evidence you need to show that you made a good decision?

Augmenting real-time V&V

In order to live up to expectations AI implementations need both a high quantity and very high quality of data inputs. RKVST can feed AI on known good sources and prove to others it made good choices, building confidence in the decisions and recommendations made by AI engines to boost enterprise adopters far beyond competitors on their digital transformations.

By enabling confident exchange of data, the RKVST platform is already a huge leap forward for connected industries and digital transformation by creating a single, accessible, consolidated picture of the service history of any asset, and by extension its trustworthiness.

Beyond exchanging single pieces of asset data (such as current location, alarm status etc.) the collective service history information can be used to identify important issues such as cyber vulnerabilities, maintenance errors, or handling policy violations. And because of the flexible JSON document structure at the heart of Asset records clients can write interrogation code to read this history and validate the parts that are important to their own specific judgements without the need for the RKVST system to have semantic understanding of their data fields. This is very valuable in audit and Root Cause Analysis (RCA) situations *after* an incident, but it may be too slow and cumbersome for real-time decisions that might *avert* an incident.

In order to assist with incident aversion and real-time V&V, RKVST features a capability called "compliance posture" which takes the processing burden off the client by providing a single, simple API call to answer the complex question: "given all you know about this asset, should I trust it right now?". Additionally, and crucially for sensitive use cases, the yes or no answer comes with a detailed defensible reason why, which can be inspected by relevant stakeholders (including AI engines or compliance software) during or after the event.

As an example, compliance posture policies can answer questions such as "Has my device remained unpatched for more than 30 days from a vulnerability report?" or "was the safe handling process carried out in the right order and with the right timing?" or "is a newer approved configuration available than the one I'm running?"

This simple interface to answering hard problems allows real-time V&V of documentation, device configurations, or devices themselves, in order to make informed and defensible decisions about whether to proceed with a digital operation.

As an example, it is well known by now that "things are only secure until they're not", and any software and hardware in the field for long enough will have exploitable vulnerabilities. This makes existing simple practice for security rather fragile: even if a device has a hardware key and a production certificate, it may still have a software bug that enables rogue code to take it over. At that point things get a lot worse, since the rogue code is able to use that key to interact with the rest of the system, and many security products will let it straight through because of the supposedly strong identity of the device. Figure 14 below illustrates an Identity and Access Management (IAM) platform using RKVST compliance posture checks to make dynamic trust decisions for IoT devices that are trying to access a network resource. This capability helps to defend against certificate errors, lost keys, and compromised firmware, moving significantly towards a 'Zero Trust' model for IoT. The IAM vendor has been able to adopt the compliance posture APIs to perform additional checks on IoT devices accessing enterprise resources. Instead of simply checking the production certificate and letting the request through, the IAM platform additionally checks the compliance posture of the device, for things like 'firmware up-to-dateness'. If the check with RKVST fails then the platform is able to respond appropriately, either by refusing access or going into a step-up path or allowing degraded access.

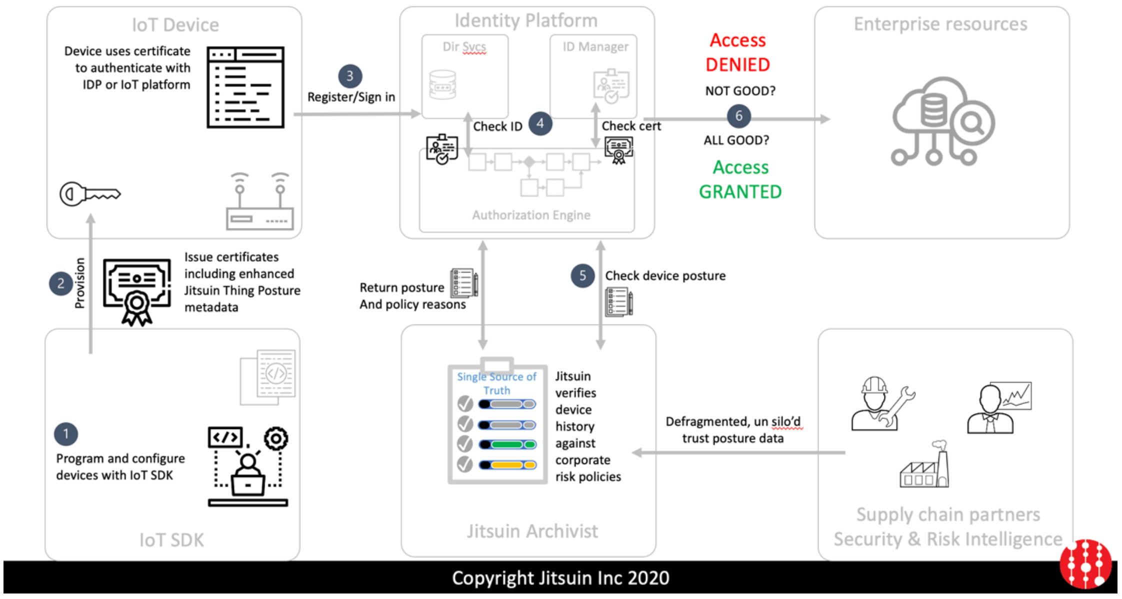

IoT Device
Device uses certificate to authenticate with IDP or IoT platform
3
Register/Sign in
Identity Platform
Dir Svcs
ID Manager
Check ID
4
Check cert
Authorization Engine
Access DENIED
NOT GOOD?
6
ALL GOOD?
Access GRANTED
Enterprise resources
2
Provision
Issue certificates including enhanced Jitsuin Thing Posture metadata
Return posture And policy reasons
5
Check device posture
1
Program and configure devices with IoT SDK
IoT SDK
Single Source of Truth
Jitsuin verifies device history against corporate risk policies
Jitsuin Archivist
Defragmented, un silo'd trust posture data
Supply chain partners Security & Risk Intelligence
Copyright Jitsuin Inc 2020

Figure 14: Towards 'zero trust'. Compliance posture based dynamic trust decisions for IoT

Note that at present this compliance posture functionality is limited to identifying time gaps and maintenance event process flow in Asset records (providing insight to Service Level Agreement (SLA) or Maintenance, Repair, and Operations (MRO) failures) or vulnerability exposure windows (providing insights into responsible cybersecurity management). Jitsuin is heavily investing in the development of this functionality to enable much more valuable digitalization and automation use cases through a feature which we call "Explainable Trust for AI". Please note that these are roadmap items and no promise can be made as to the timing of availability, but by the end of 2021 the platform is expected to have the following improvements:

- Add *richness* to compliance policies so that they can include event and asset data. For instance: "do not accept this shipment if its rad level is greater than 7."
- Add *dynamic tolerance* to compliance policies for more powerful contextual decision-making. For instance: "do not accept this shipment if it has been delayed more than 10% of average time."
- Add *perspectives*: In audit activities it is important to know "who knew what when?" or "is it reasonable to expect the parties to have known there was a problem back then?". RKVST service histories can already answer this question through linear analysis. We intend to add additional automation to the compliance APIs that allow stakeholders to ask the retrospective question: "regardless of compliance posture now, given the data in the system at the time, would a trust check have passed or failed on date X, and why?"
- Add *semantic interpretation* to compliance policies so that they can include interpretation of event and asset data. For instance: "do not accept this shipment if its tracking data is missing", or "do not talk to this device if it has ever been seen outside the United Kingdom."

With all this done, AI automation becomes much more trustworthy and explainable in the real world: not only does the AI have a greater range of more trustworthy data to work with, but external stakeholders can also verify that they are being operated responsibly and machine errors can be thoroughly investigated.

Technical Interface

API

RKVST presents a simple REST API to fetch and manipulate the Asset and Event JSON. A full reference is publicly available at https://jitsuin-archivist.readthedocs.io/en/latest/

Data structures

Asset and Event records are extensible JSON documents, containing some fixed elements, and any number of custom elements provided by the participants. This extensibility is crucial in supporting industrial use cases where some parts of the system, or datatypes, or processes are unique and cannot be changed easily to meet the needs of a cloud computer system or IoT platform.

Assets

```
{
  # identity is a global UID for this asset
  "identity": "assets/6a84c94e-cedb-4934-bf04-7e97d866fe73",
  # behaviours determine which smart contacts the asset can interact with
  "behaviours": [
    "Attachments",
    "Firmware",
    "LocationUpdate",
    "Maintenance",
    "RecordEvidence",
    "Builtin",
    "AssetCreator"
  ],
  # attributes is a list of tracked attributes, with complete service history.
  # elements with an 'arc_' prefix are interpreted by the system. All others
  # are completely free and expansible by the participants.
  "attributes": {
    "nda_waste_code": "42",
    # arc_attachments is a special attribute that allows the connection of
    # large binary data to the ledger record - for example, safe handling
    # manuals, photographs, or firmware
    "arc_attachments": [
      {
        "arc_display_name": "arc_primary_image",
        "arc_hash_alg": "SHA256",
        "arc_hash_value": "4afcecb8b2c1fdf5bcb487aca74831262be8b0aaeb4e69f5abc57a56db8485d4",
        "arc_attachment_identity": "blobs/7c2cb418-591d-466c-a918-756553c2a6de"
      }
    ],
    "arc_description": "An item called Oscar",
    "arc_display_name": "oscar",
    "arc_display_type": "Nuclear Waste Item",
    "nda_in_container": "assets/9410c31f-6663-44e1-89af-9f69e6d1a004",
    "nda_lifecycle_stage": "packaged",
    "nda_namespace": "test1"
  },
  # Confimation status tells whether it's finalized on the blockchain or not
  "confirmation_status": "CONFIRMED",
  # Is the asset still actively being tracked?
  "tracked": "TRACKED",
  "owner": "0xe3B38F2aA6a0823178939eEDC70E50FFFF1e83E1"
}
```

Events

```
   {
      # identity is a global UID for this event
      "identity": "assets/6a84c94e-cedb-4934-bf04-7e97d866fe73/events/af79feb5-c09d-459b-818a-
1e979c43259b",
      # foreign key to the identity of the asset it applies to
      "asset_identity": "assets/6a84c94e-cedb-4934-bf04-7e97d866fe73",
      # event_attributes is a list of attributes fo this event ONLY – a record
      # of what happened, but does not update any other system or Asset state.
      # Elements with an 'arc_' prefix are interpreted by the system. All others
      # are completely free and expansible by the participants.
      "event_attributes": {
        "arc_display_type": "Characterize",
        "arc_evidence": "No evidence provided",
        "arc_description": "Calculated and applied fingerprint '42' to item"
      },
      # asset_attributes is a list of Asset attributes that this event updates.
      # The Asset record will be updated with these values, and the service
      # history will show the evolving state of the Asset.
      # Note 1: This is access controlled. The principal MUST have an access
      # policy permission to wite this attribute value.
      # Note 2: If this property does not yet exist on the Asset, it is created
      "asset_attributes": {
        "nda_lifecycle_stage": "characterized",
        "nda_waste_code": "42"
      },
      # Which contract/API was called?
      "operation": "Record",
      "behaviour": "RecordEvidence",
      # Timetamps
      "timestamp_declared": "2021-03-23T23:43:58Z",
      "timestamp_accepted": "2021-03-23T23:43:58Z",
      "timestamp_committed": "2021-03-23T23:44:00Z",
      # Who registered the event and where did RKVST validate it against?
      "principal_declared": {
        "issuer": "https://login.microsoftonline.com/dc229635-5858-4fe3-9bef-
444f6c7ee1cf/v2.0",
        "subject": "0dad5d0c-35e0-49f0-9381-0a8331e2efa9",
        "display_name": "0dad5d0c-35e0-49f0-9381-0a8331e2efa9",
        "email": ""
      },
      "principal_accepted": {
        "issuer": "https://login.microsoftonline.com/dc229635-5858-4fe3-9bef-
444f6c7ee1cf/v2.0",
        "subject": "0dad5d0c-35e0-49f0-9381-0a8331e2efa9",
        "display_name": "0dad5d0c-35e0-49f0-9381-0a8331e2efa9",
        "email": ""
      },
      # Blockchain details, for shared and mutually distrustful verification
      # of the contents and timestamp of the Event for ANY participant.
      "confirmation_status": "CONFIRMED",
      "transaction_id": "0x5d135800554ff8a9029e302ff88446be0be6c56e49a7b268bbf78d94ea1463f6",
      "block_number": 82453,
      "transaction_index": 0,
      "from": "0xe3B38F2aA6a0823178939eEDC70E50FFFF1e83E1"
    }
```

User Authentication

Users, devices, and client software can authenticate themselves using one of 2 methods:

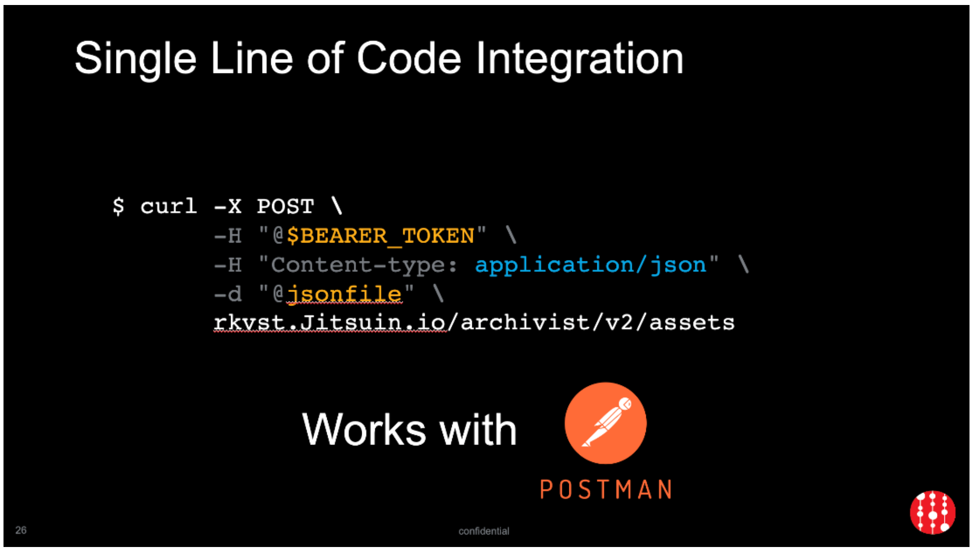
- A bearer token, in the form of a JWT authorized by one of their tenant's configured identity providers
- Mutual TLS, using a client key signed by a CA that is trusted by their tenant

Each individual command must carry user authentication: there are no 'logins' or 'sessions'. This improves system resilience and simplifies client integrations by eliminating system-wide state, but more importantly it enables the system to validate each read and write operation against up-to-date sharing policies.

Note: The configuration surface of RKVST allows reference to users by friendly attributes such as name and email, but in the underlying system users are always identified and validated by their unique issuer:subject pair which ensures a clear cryptographic link back to their corporate ID system.

Low-code and Zero-code integration

As demonstrated in Figure 15 below, being based on stateless REST interfaces and having flexible authentication and message structures, RKVST supports zero-code integration with client systems and assets. A zero-code cURL one-liner can be used to create a fresh Asset record, given a bearer token and a JSON document containing the Asset properties. It is *possible* to create a client-side agent that links a secure-by-design hardware chip identity and strong message signatures to the events in the Asset record, but it is not *necessary*.



Single Line of Code Integration

```
$ curl -X POST \
       -H "@$BEARER_TOKEN" \
       -H "Content-type: application/json" \
       -d "@jsonfile" \
       rkvst.Jitsuin.io/archivist/v2/assets
```

Works with
POSTMAN
26
confidential

Figure 15: Zero-code cURL one-liner to create a fresh Asset record.

Use of RKVST for ResiCAV+

With all this known, the use of RKVST to support ResiCAV+ is simple:

- Model the ideal baseline configuration as an asset (A_M)
  - Any and all of the authorized participants (both humans and software agents from Warwick, Thales, Cardiff and Bristol) can then update the ideal model as learning or development proceeds. This can be achieved by a small REST call being added to the existing build processes and tools.
  - Every observer can see how the model emerges.
- Model every platform as an Asset (A_{Pi}) and store its configuration as Asset attributes.
  - Include significant difference identifiers and validation as access-controlled attributes
  - Add a REST call hook into the update server, to register the intent to update and the proof of why this is a good update to make. That proof can include a call to RKVST to compare the contemporary state of A_M with the new baseline.
  - Add a REST call hook to the platform update code, to say where it got its update from, what the update looked like, and whether it applied successfully. The platform

can also perform its own realtime V&V by verifying the trustworthiness of the originator of the update and the current state of A_M before accepting the update.

- For Pattern of Life and Ordering assurance, the users can simply increase the number and granularity of Events registered
- Regulators can also participate (possibly in a read-only capacity) and survey the entire estate for real-time compliance, process, and security, and can rely on having their own copy of the evidence base without any need to request extra docs o access to participant databases and systems.

Meeting the '3 tests'

The purpose of the ResiCAV+ program is to improve the *useful* quality of electronic evidence gathering, as defined by 3 tests:

- The methodology is capable of being tested in court or by publicly appointed regulators.
- Operators understand what evidence should be produced by it and are able to measure the quality of that evidence.
- The evidence produced is capable of being tested in court or by publicly appointed regulators.

The methodology is capable of being tested in court or by publicly appointed regulators.

While "the methodology" includes many things outside of the RKVST system, the RKVST architecture and workflows support this goal by linking each participant's RKVST identity and access configuration directly to their own corporate Single Sign On and by extension to their existing testable corporate compliance standards. Therefore, issues of digital representation, control etc are easily examined and understandable.

Operators understand what evidence should be produced by it and are able to measure the quality of that evidence.

The flexible JSON document format of RKVST enables a tight and faithful correlation between the data in RKVST Asset histories and the data that each operator naturally understands and works with on their local/native systems and processes. Therefore, operator understanding is readily achieved.

Evidence quality assessment is supported by two of the main features of RKVST:

- The high integrity features which ensure that Events cannot be backdated, tampered with, or removed.
- The fair access that ensures every authorized participant has access to their complete data set without the need to request access to other people's databases.

The evidence produced is capable of being tested in court or by publicly appointed regulators.

In cases where the regulator is known and constant, they may have their own RKVST tenancy and review the evidence shared with them at any time: even ahead of any court action.

In other cases, the regulator can easily gather evidence from every participant and quickly compare them for integrity against each other and against the shared blockchain record. Any discrepancy is quickly identified since attempts to hide or change Event data will show up when comparing the rich evidence to the blockchain.

Furthermore, a common problem with RCA and testing disparate data sets in court is comparing timestamps: how to consolidate many fragmented logs if you can't be sure what happened when, or whose clock was more accurate? With RKVST (and blockchain systems in general) the timestamps and ordering are inherently aligned, making analysis fast and reliable.

A Note on Digital Twins[3]

Dynamic Resilience

The DTC security and Trustworthiness group focuses on "Dynamic Resilience". Many things can change after the initial design of a component or digital twin system: software bugs may be discovered, compliance standards may change, physical faults may develop, operator training standards may slip…it is not knowable up-front every possible failure mode or their consequences. Given this, a heavy focus on one technical aspect of security may prove ineffective against emergent threats or environmental factors and check-box static compliance policies will fail to find to prevent problems.

Additionally, Digital Twins only exist in a DX context where a Physical Twin is connected. Physical infrastructure evolves much more slowly than the virtual, and so many incremental system upgrades and component swaps can be expected over the life of a Digital/Physical twin pair. Each time this happens, the design files, security assumptions, provenance data and operating models of the new software and hardware pieces need to be folded into the existing twin system and the entire system needs to be re-certified and re-tested against its safety models.

Such re-certification is a well-known part of Configuration Management and Systems Engineering disciplines. But typical CM/SE processes are heavily manual and require a lot of cross-checking. The data in them is silo'd and pulling sufficiently high-quality supply chain data to add to the evidence/assurance base is hard. These processes simply cannot scale to the complexity and pace of change that DX demands. Automation based on trustworthy operating and security data from partners is essential.

Most Digital Twin designs do not yet include Dynamic Resilience, nor do they often feature the kind of process traceability.

Frequency and Fidelity

> *"A Digital Twin is a virtual representation of real-world entities and processes, synchronized <u>at a specified frequency and fidelity"</u>*
>
> \- *Digital Twin Consortium definition*

No matter how good Digital Twin models are, they will have limitations. It is important to understand the quality and timeliness of the connection between a sensor and the data it produces in order to understand how trustworthy the twin might be for a particular use case. Accurate data showing up 1 day late might be just as bad as inaccurate data arriving every millisecond. Or they might be fine. But that call can only be made if the frequency and fidelity of synchronization is known and transparent for all sources, both internal to the owning organization, and external.

[3] *The author of this section of the report presently serves as Chair of the Security and Trustworthiness Working Group in the Digital Twin Consortium and actively contributes to the development of standards and best practices for trustworthy Digital Twin deployments*

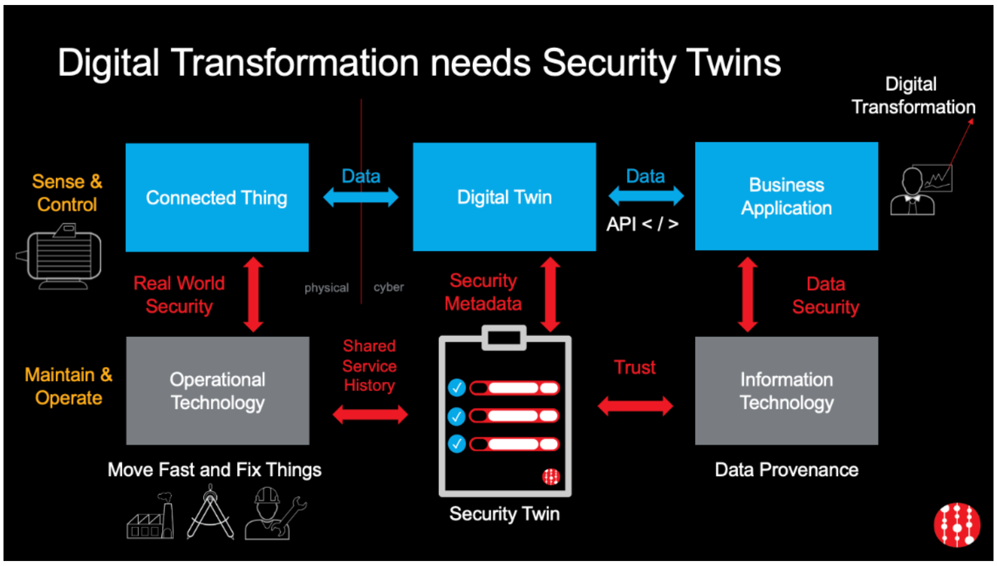
And of course, don't forget the source of the synchronized data also needs to be very keenly understood. Without additional safeguards if a rogue operator puts their hand over a thermometer, it will very faithfully report a wrong temperature.

Environmental Context

Some important factors to security and safety belong to no particular organization and are not necessarily electronic or digital in nature, so they stand no chance of being included in any individual Digital Twin or Digital Twin Model. And yet they need to be taken into account if the twin is to be trusted with automation. Whether an active attack, or an accident, a component recall or a freak weather event, the operating context must be taken into account.

Use of ledger-based technologies to address these problems

A ledger is a very effective way of solving all these issues, for example by allowing threat intelligence to be reported easily once by a reptable source and retrieved wherever it's needed, and by tracing all of the necessary dynamic data that is needed for the execution of effective dynamic assurance cases.



For a more in-depth treatment on this topic, please refer to this Cutter Business Technology Journal article by the same author: https://www.cutter.com/offer/security-trustworthiness-digital-twin-systems

Appendix C: The Significant Difference Tool

Background

Motivation Diversity inspired Defence Strategy

- Originally used to indicate the sustainability and survivability of an ecosystem
- Diversified components make the overall system resistant against sudden changes, faults and attacks
- Digitally identical components fail at the same time giving rise to catastrophic failure.

Diversity inspired strategy studied since the 1970s to enhance the security and resilience.

- N version programming, program obfuscation, code randomization, etc.
- Diversifying routing nodes, software packages, OS, etc.
- Antenna diversity for key generation, architectural diversity for FPGA, etc.

Previous work: Attack Graph - Based Diversification Tool[4]

➢ Generate diversified deployment when upgrading ICS with dynamic IT systems.

➢ Statistically measure the vulnerability similarity between software based on CVE database.

➢ Adapting to various changes and constraints in ICS.

What's missing

➢ Difficult to determine one component is significantly different from another.

➢ Lack of a valid diversity metric and generic way to measure diversity.

➢ Evaluate resilience brought about by diversification.

Work Package Aims

➢ Focusing on exploring structural similarities and interdependency between components.

➢ Quantifying diversity by similarly vulnerable structures (i.e. vulnerable primitives) of components.

➢ Effectively evaluate human-input diversification strategies prior to deployment.

[4] T. Li, C. Feng and C. Hankin, "Scalable Approach to Enhancing ICS Resilience by Network Diversity,"*50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2020, pp. 398-410,

Appendix D: Machine Learning Classifier Predication Tool Using Dissimilarly Measures

Object and signal classifiers can be regarded as one of the key components in Autonomous Systems. For example, robust obstacle detection is an important step in allowing any unmanned vehicle to be aware of its environment and make safe and effective navigation decisions. However, in order to assist in providing robust and safe path planning and collision avoidance, detection and tracking of objects has great impact on the vehicle's understanding and identifying of safe routes. Our confidence in the expected performance of the classifier, given information about the system environment, is an essential element of the overall system trustworthiness.

This work applied new techniques developed in Thales UK to enhance the V&V processes in complex autonomous systems, produce the evidence and give confidence in the use of Artificial Neural Network (ANN) classifier's in terms of adequacy and their accuracy, grounded in Thales use cases. The verification process contributes to the safe operation of vehicles and the safety of people.

The objectives for this work were:

- To establish the user/application requirements for classifiers
- To select and train classifiers targeted for specific applications
- To verify and predict classifier performance (an integrated process) for real-time operations.

Scope

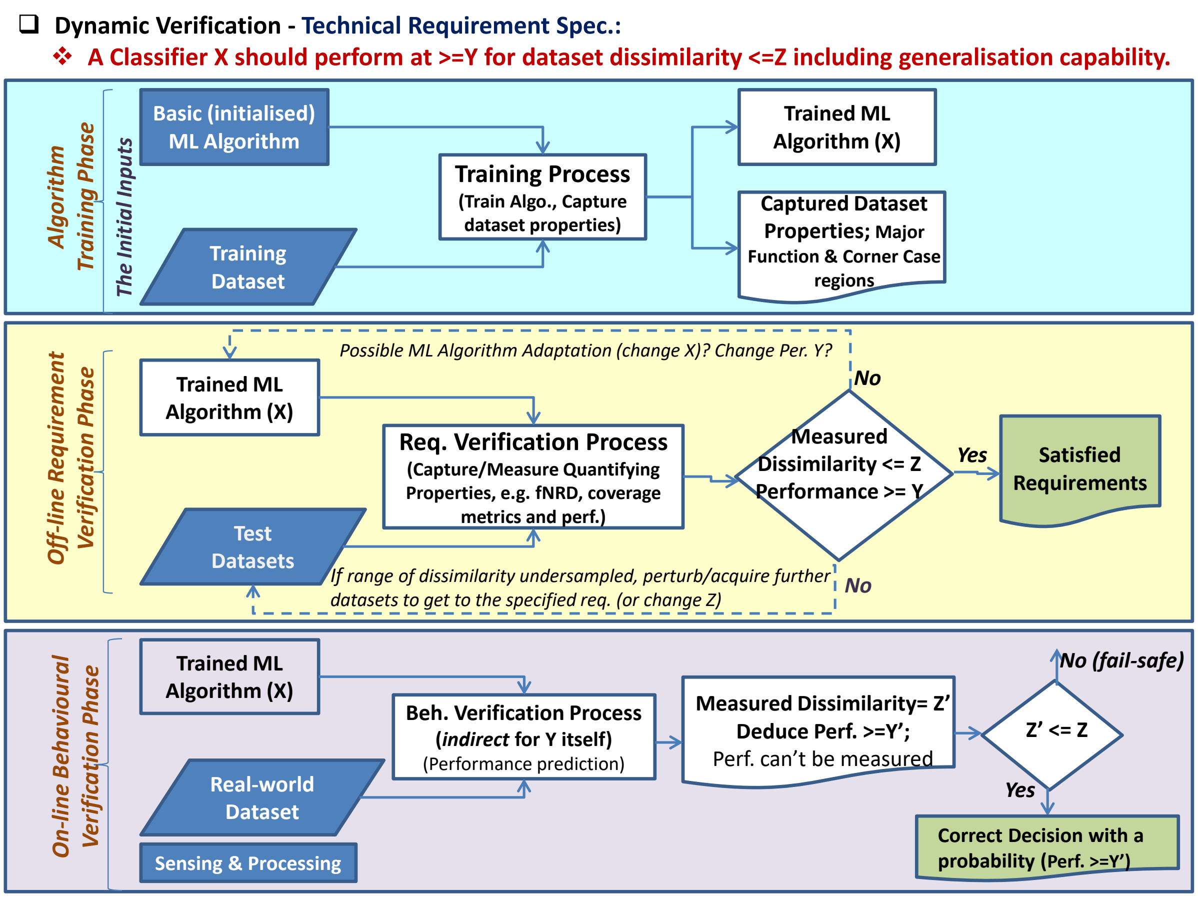
This tool addressed several challenges for targeted use-case applications and demonstrated the following features:

- A method for classifier performance prediction during operation
- A method for self-verification during operation where the related requirement specifies a permissible range of values for the domain of the classifier resilience function
- The verification of ANN classifiers against requirements for the targeted application
- Dissimilarity and test coverage measures for the verification process
- Specification of ANN classifier requirements as part of the verification process so that they state the permitted forms of multiple classifier resilience functions (i.e. stating the required generalisation capability of an ANN classifier for each quantifiable dataset property

Technical Approach

The approach was to provide a means of systematically verifying the behaviour of ANN classifiers that use real-world imagery input for a targeted use-case application. Methods were established to verify the correctness, performance, and behaviour of the classifiers. There was an emphasis on characterising the properties of datasets, and the ability of classifiers to generalise over these properties. Detailed or component classifier requirements had to be specified and verified in line with broader requirements.

The process for this is provided in **Error! Reference source not found.**. We have defined a Classifier R esilience Function (CRF) in order to indicate to what extent the classifier performance is maintained as it is tested on a series of datasets which, by some measure, are progressively further from the dataset on which the classifier was trained. The CRF indicates the ability of the classifier to generalise to more distant test datasets. More generally, the domain of the function could be any quantifiable property of test dataset dissimilarity.



Dynamic Verification - Technical Requirement Spec.:
A Classifier X should perform at >=Y for dataset dissimilarity <=Z including generalisation capability.
Algorithm Training Phase
The Initial Inputs
Basic (initialised) ML Algorithm
Training Dataset
Training Process (Train Algo., Capture dataset properties)
Trained ML Algorithm (X)
Captured Dataset Properties; Major Function & Corner Case regions
Off-line Requirement Verification Phase
Possible ML Algorithm Adaptation (change X)? Change Per. Y?
No
Trained ML Algorithm (X)
Req. Verification Process (Capture/Measure Quantifying Properties, e.g. fNRD, coverage metrics and perf.)
Measured Dissimilarity <= Z Performance >= Y
Yes
Satisfied Requirements
Test Datasets
If range of dissimilarity undersampled, perturb/acquire further datasets to get to the specified req. (or change Z)
No
On-line Behavioural Verification Phase
Trained ML Algorithm (X)
Beh. Verification Process (indirect for Y itself) (Performance prediction)
Measured Dissimilarity= Z' Deduce Perf. >=Y'; Perf. can't be measured
Z' <= Z
No (fail-safe)
Yes
Correct Decision with a probability (Perf. >=Y')
Real-world Dataset
Sensing & Processing

Figure 16: Classifier based on Training, Offline and Online Verification.

Figure 16 illustrates the three phases of quantifying the properties and verification of classifier performance. Here, off-line refers to the training and testing phases and on-line refers to the real world operation of the classifier. Classifiers are trained using real data captured by sensors, or using synthetic data. This data forms the training dataset and test dataset. Classifiers are put through performance tests using unseen test datasets. Performance on a test dataset is measured using some metrics developed at Thales UK. By some measure how far or how separate is the test dataset from the training dataset. Measuring and quantifying test dataset properties allow the level of performance to be expressed as a function of these properties for a given classifier selected for a specific use-case application. The classification performance is qualified by specifying to what extent the test dataset will challenge the classifier. During an autonomous system's operation, the "dissimilarity" measure can be calculated using datasets captured on-line, that is to say sets of images captured by the system over methodically determined time intervals.

As shown in the off-line requirement verification phase in Figure 16, during initial requirement verification there are possibilities to re-train the classifier, change the classifier, change the performance demand based on the application and the range of dissimilarity addressed, prior to putting the classifier into operation.

Work Structure

This work is to use the tool developed at Thales UK to generate concrete test cases for evaluation of autonomous vehicles (AV). It includes:

- The use of Scenario Description Language (SDL)
    - SDL describes the static content (e.g. road network) and dynamic content (e.g. vehicle manoeuvers) of AV test scenarios

    - Start with high-level, abstract scenario description, i.e. human readable (interpretable by various stakeholders, e.g. regulators, users, developers, testers)

- Generation of executable tests for execution in a synthetic environment; Many concrete scenario descriptions are generated from an abstract scenario description by varying parameter values (e.g. vehicle speed, weather, road layout)

Tool Chain

The architecture diagram in Figure 17 illustrates some of the key components and interfaces to the Thales UK's developed tool chain.

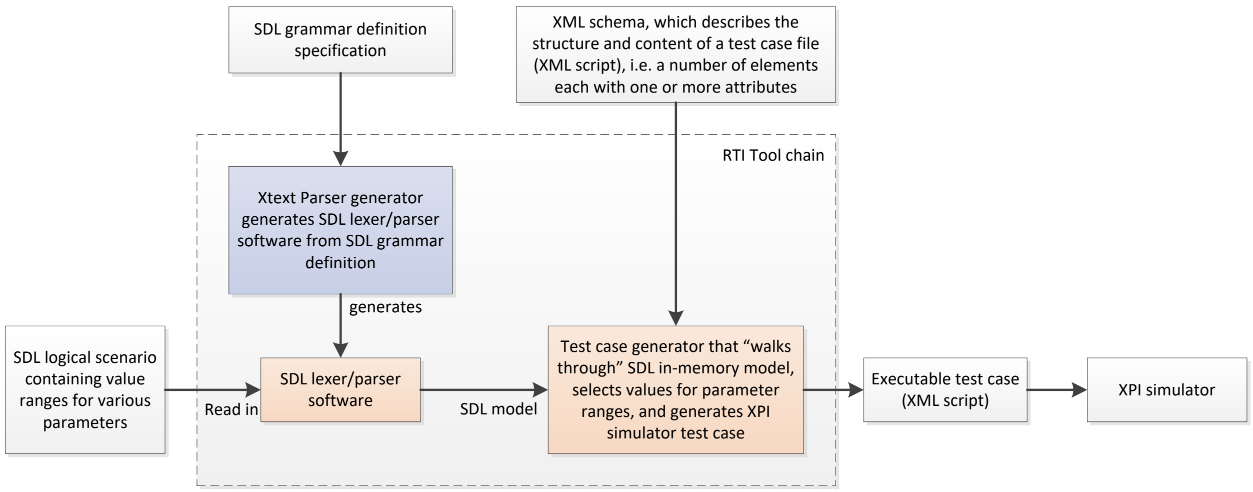
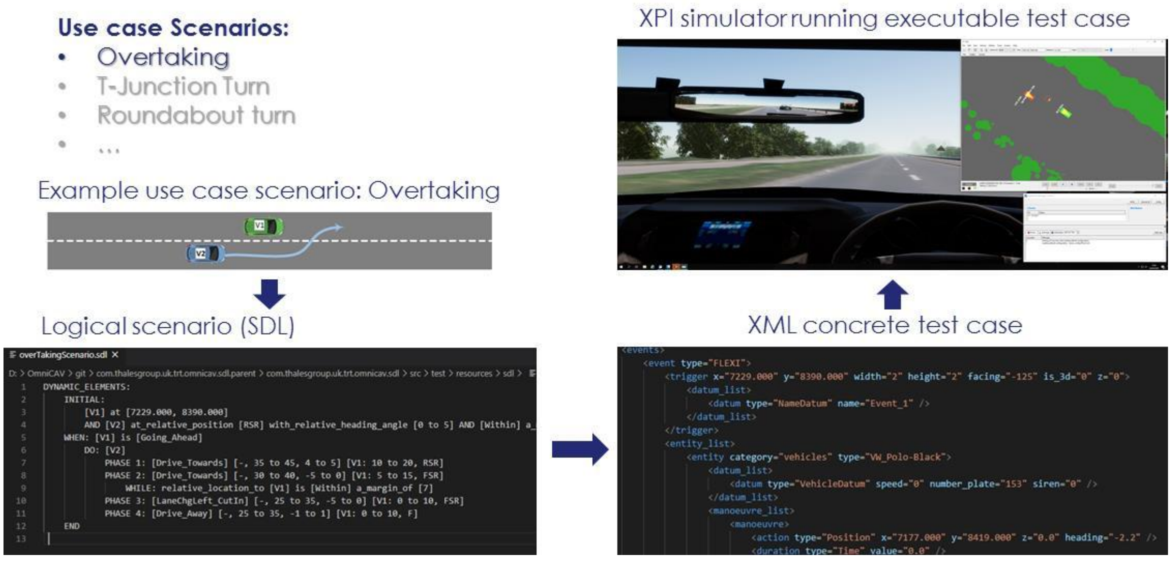

SDL grammar definition specification
XML schema, which describes the structure and content of a test case file (XML script), i.e. a number of elements each with one or more attributes
RTI Tool chain
Xtext Parser generator generates SDL lexer/parser software from SDL grammar definition
generates
SDL logical scenario containing value ranges for various parameters
Read in
SDL lexer/parser software
SDL model
Test case generator that "walks through" SDL in-memory model, selects values for parameter ranges, and generates XPI simulator test case
Executable test case (XML script)
XPI simulator

Figure 17. Architecture for automated generation of executable test cases for SDL.

Figure 18 shows the tool chain developed by Thales UK.


Use case Scenarios:
Overtaking
T-Junction Turn
Roundabout turn
...
Example use case scenario: Overtaking
V1
V2
Logical scenario (SDL)
XPI simulator running executable test case
XML concrete test case
overTakingScenario.sdl
D: > OmniCAV > git > com.thalesgroup.uk.trt.omnicav.sdl.parent > com.thalesgroup.uk.trt.omnicav.sdl > src > test > resources > sdl >

```
DYNAMIC_ELEMENTS:
    INITIAL:
        [V1] at [7229.000, 8390.000]
        AND [V2] at_relative_position [RSR] with_relative_heading_angle [0 to 5] AND [Within] a_
    WHEN: [V1] is [Going_Ahead]
        DO: [V2]
            PHASE 1: [Drive_Towards] [-, 35 to 45, 4 to 5] [V1: 10 to 20, RSR]
            PHASE 2: [Drive_Towards] [-, 30 to 40, -5 to 0] [V1: 5 to 15, FSR]
                WHILE: relative_location_to [V1] is [Within] a_margin_of [7]
            PHASE 3: [LaneChgLeft_CutIn] [-, 25 to 35, -5 to 0] [V1: 0 to 10, FSR]
            PHASE 4: [Drive_Away] [-, 25 to 35, -1 to 1] [V1: 0 to 10, F]
    END
```

```
<events>
    <event type="FLEXI">
        <trigger x="7229.000" y="8390.000" width="2" height="2" facing="-125" is_3d="0" z="0">
            <datum_list>
                <datum type="NameDatum" name="Event_1" />
            </datum_list>
        </trigger>
        <entity_list>
            <entity category="vehicles" type="VW_Polo-Black">
                <datum_list>
                    <datum type="VehicleDatum" speed="0" number_plate="153" siren="0" />
                </datum_list>
                <manoeuvre_list>
                    <manoeuvre>
                        <action type="Position" x="7177.000" y="8419.000" z="0.0" heading="-2.2" />
                        <duration type="Time" value="0.0" />
```

Figure 18. Architecture for automated generation of executable test cases for SDL.

Appendix E: A Tool For AI Driving of Real Time V&V

Introduction

The CyRes methodology has been demonstrated as a possible solution to mitigate against cyber-attacks in the face of autonomous and cyber-physical systems deployed on a mass scale which may be vulnerable to a single attack.

In this section we demonstrate how part of the CyRes pipeline was realised using the techniques and methods of online simulation-based verification that researchers at the University of Bristol's (TSL) Trustworthy Systems Lab have experience in.

Objective

CyRes is an innovative engineering method for operational cyber resilience. It aims to enable cyber resilience against vulnerabilities of mass-adopted cyber physical systems. One approach to achieve this is the introduction of significant difference in otherwise digitally identical individual systems. Traditional production processes of digital systems are designed to keep what is produced as similar as possible to leverage economies of scale. This makes all digitally identical systems vulnerable to the same cyber-attack, potentially at the same time. The deliberate introduction of significant difference aims to counter-act this weakness.

Thus, significant difference will be introduced to prevent an epidemic-type response to a specific cyber-attack against a large and complex cyber-physical system. For example, this could occur against a fleet of CAVs (Connected and Autonomous Vehicles). Any difference that is introduced may have unintended consequences that might, for example, compromise the safety or functionality of the overall system.

The expertise at TSL provided insight into the properties of the proposed differences by utilising advanced verification techniques and environments. In particular, runtime verification techniques employed to assess whether the proposed differences give rise to unintended functional or safety related changes in system behaviour.

Part of the CyRes process pipeline is shown in Figure 19. The process begins with the identification of system instability through a monitoring process. This stability monitoring identifies an instability in the system which may then trigger a request for a significant difference to be generated. The significant difference generation process would identify a difference or set of differences. This set of candidate cases can then be analysed by an online simulation-based verification process which would seek to ensure any candidate design proposal would result in functional equivalence with the original, including safe operation. This candidate design can then be passed on to the next stage in the pipeline.

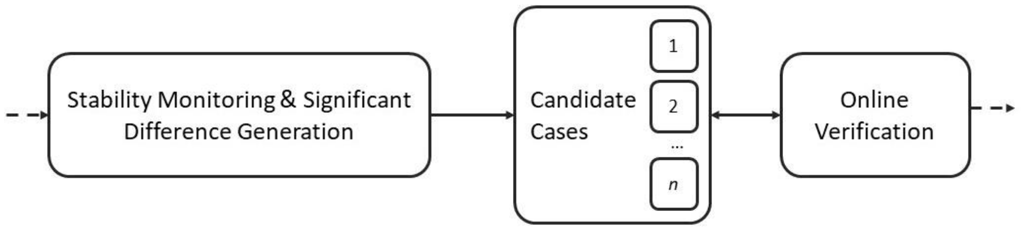


Figure 19 A section of the CyRes process pipeline.

Process in detail

To illustrate the process further we take the case study of a vehicle braking system. Prior to the online verification in the pipeline, the stability of the system is monitored for a potential cyber-attack. Figure 20 is an example of some of the possible data channels required for monitoring the braking system. Key data channels for the braking system may include velocity, acceleration and steering angle of the vehicle which are internal sensors to the vehicles.

Additionally, there may be data channels that exist partially or entirely external to the vehicle. These may include d_{v2v} (the distance to the leading vehicle), messages between the vehicle and the local infrastructure (V2I) and location tracking such as GPS. These signals will be carried around the vehicle on a bus network, which is commonplace in modern vehicle systems.

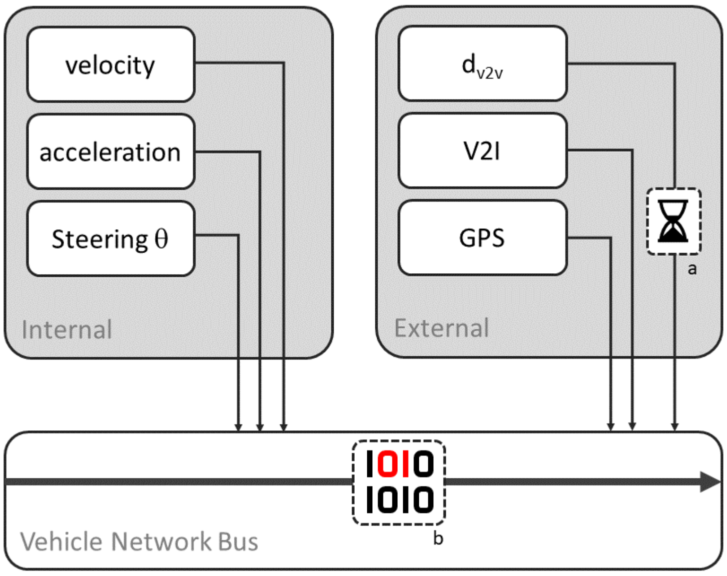


Figure 20 An example of internal and external data signals arriving on a vehicle bus.

The data channels used in the braking system shown in Figure 20 may be susceptible to attacks on the communication between subsystems on the network such as a variety of Electronic Control Units (ECUs) serving different in-vehicle domains. Two types of attack are used in this example; where a delay on a signal reaching the bus is introduced and, where packet loss occurs in a signal. The attacks in this case study are; a delay in the d_{v2v} signal shown at Figure 20 (a) and missing data or packet loss shown at Figure 20 (b). Such attacks can be easily modelled, and the stability analysis tool can be used to discern such an attack from general network traffic or signal noise.

Attacks may result in changes in the observable behaviour of the system. For example, a delay to the d_{v2v} signal reaching the vehicle control system may result in an improper distance being kept to the leading vehicle in a driving situation where the autonomous vehicle is following another road user. A delay on other signals, for example the steering angle may result in disruption to the control algorithms that maintain the vehicle in the centre of the lane. Both of these externally observable behaviours can be monitored and assessed independently of the observations within the vehicle network bus. Simulation can be used to assess the safe operation of the vehicle based on its externally observable behaviour.

Simulation-based Verification at Runtime

The use of simulation in the CyRes methodology is manifold:

- Simulation can be used to help identify potential threats using online behavioural monitoring in close combination with a stability monitoring process.
- By identifying potential attack vectors (where in the vehicle an attack is targeting), simulation can be used to determine whether the vehicle has been brought back to safe operation following a mitigation action.
- Simulation can provide insight into the potential impact of an attack once an attack has been identified.
- Simulation can also be used to assess whether the application of design changes to achieve significant difference result in functional and safety-related equivalence in the modified system.

Although there is merit in all these uses of simulation, this report will focus on the last point in the following sections as this pertains directly to the CyRes methodology pipeline under consideration.

Automated Test Generation

Test case generation forms an integral and necessary part to the simulation aspect of the toolchain. Without suitably defined test cases the simulator cannot operate. Test cases provide stimulus to the simulation and can be used to drive the simulation towards certain conditions of interest with respect to verification. For example, if pedestrian safety is part of the verification task, then test cases should guide the simulation toward scenarios involving pedestrians to assess CAV behaviour. The behaviour of the CAV can be witnessed in simulation and is written into logs for checking. The evidence so gained could then be presented if and when required by regulators or legal challenges.

To identify suitable automated test generation techniques, it is first necessary to describe the verification process that will be called upon and how the simulation results can be interpreted into simple pass or fail categories. An example testbench is shown in Figure 21.

Simulation-based Verification

The use of simulation aligns well with the CyRes approach, where simulation can be used to test variants of the system design for functional equivalence and safety prior to applying a change of significant difference.

At the centre of the testbench is the simulator, which needs information about the static and dynamic elements that will be needed at runtime. For the braking scenario this may include static objects such as a road with line markings, and dynamic objects, or actors, such as the autonomous vehicle under test and a vehicle that it is following. These static and dynamic objects will be called from the experiment instantiation which will convert the experiment specification to a machine readable and executable form. The experiment specification contains part of the information needed in the test case and these will be driven by requirements. Coverage can be defined based on different aspects, as shown in Figure 21. Coverage models include requirements coverage to ensure tests cover the original requirements, implementation-specific coverage to ensure all code in the system has been tested, i.e. different forms of code coverage, and also coverage to confirm that all functionality has been observed, i.e. functional or assertion coverage.

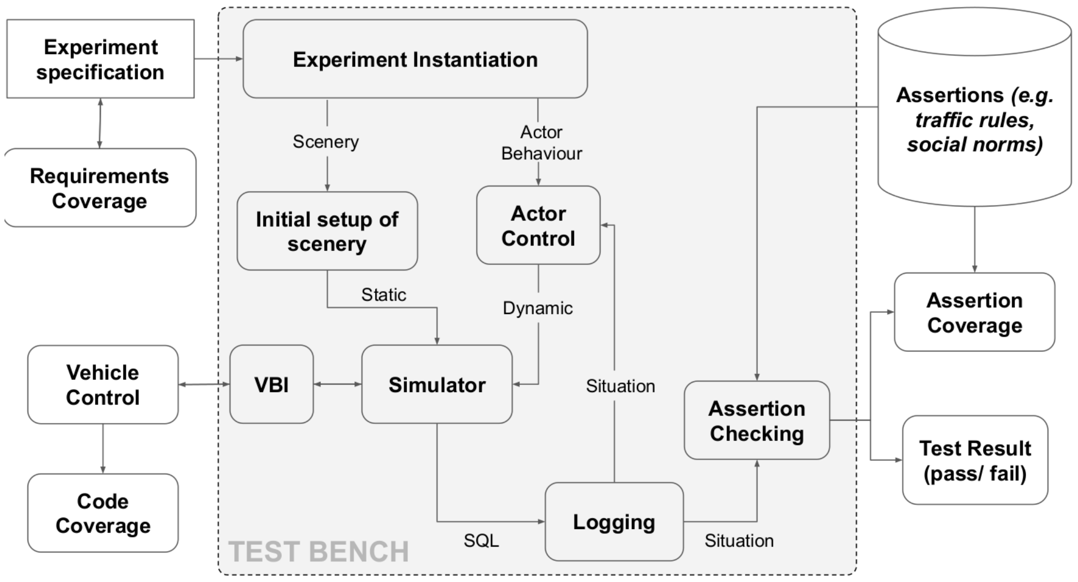


Figure 21 Example testbench for simulation-based verification of autonomous vehicles.

In the use case described here, the requirements will need to specify the safe operation of the vehicle following a significant difference made to the system. It is therefore necessary to have test cases that:

- Demonstrate the changes made do not adversely interfere with the safe operation of the vehicle,
- Demonstrate that no unintended consequences that may impact on the safe operation of the vehicle have been introduced into other aspects of subsystems as a consequence of the changes, i.e., if there was a change in the speed monitoring that this change did not adversely impact other aspects of the vehicle dynamics such as acceleration.

Moreover, if test cases are to be generated automatically to align with the CyRes framework then these should follow the principles of a 'good test case' by Fewster and Graham[5] which are shown in Table 1. Test cases should be effective in finding bugs and more specifically to CyRes, should be effective in determining if the significant difference made to the system will result in unsafe operation. Considering the manner of deployment for CyRes, the principle should be to monitor the system during operation and therefore any test cases online may need to find bugs or issues with the changes efficiently, i.e., by executing as few test cases as possible. Test cases should also use resources efficiently by minimising simulation cycles and leveraging the use of efficient computational libraries.

Robustness is the final criterion in the list of qualities for a good test case which warrants a dedicated explanation. Test case robustness is the quality associated with reuse following a software or system change. If a test case is generated but is also effective in the testing of a different system, then it becomes highly effective. This is of particular importance to the CyRes method, as the fundamental approach will be to simultaneously test sets of candidate cases with significant difference applied. If a single or small number of tests can be used against a larger set of candidate designs, then such test cases are highly desirable.

[5] M. Fewster and D. Graham, Software Test Automation, Addison-Wesley Reading, 1999.

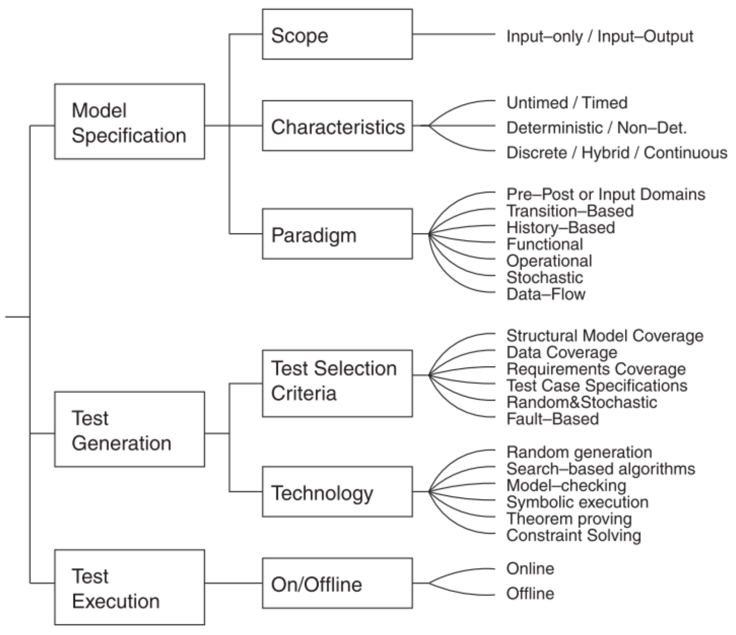
Criteria	Definition
Effectiveness	How effective the test case is in finding the failures in the system if they exist
Efficiency	Minimising the number of test cases required to achieve a level of confidence in the design change
Economy	Economically using resources such as simulation cycles, CPU hours, LOC (lines of code)
Robustness	The robustness of the test case to changes in the system or updates. This is particularly pertinent as robust test cases can be efficiently reused on subsequent design changes.

Table 1 The principles of a good test case[6].

Test Generation Methods

A good account of the taxonomy of model-based test generation methods is given by Utting[7] shown in Figure 22. A well-used method for test generation falls under the random class as this is usually considered low-hanging fruit in the initial stages of collecting coverage which is relatively inexpensive and no significant intervention from verification engineers is required. However, as the verification task continues there will come a point at which the *effectiveness* of the random method decreases and the rate of collecting coverage or bug discovery plateaus.

On reaching a plateau, other methods may be employed to reach the remaining edge cases faster than a random technique may achieve them. At this point the test generation needs to be more focused. This may include using a constrained approach to direct or bias random test generation, where the bounds of the parameter randomisation are constrained to force the test generation into certain areas. This technique may initially be effective, but ultimately becomes less *economical* for complex DUVs (Device Under Verification).



[6] M. Fewster and D. Graham, Software Test Automation, Addison-Wesley Reading, 1999.
[7] M. Utting, A. Pretschner, and B. Legeard, "A taxonomy of model-based testing approaches," Software Testing, Verification and Reliability, vol. 22, no. 5, pp. 297– 312, 2012.

Figure 22 Taxonomy of model-based test generation methods[8].

Random test generation methods are well suited to the CyRes methodology as they can be easily automated, but ultimately greater *efficiency* may be required when there are time-constraints on receiving the verification results, as is the case for runtime verification. In the spirit of a search-based approach, research at TSL suggests an additional entry into the model-based test generation taxonomy, namely agent-based test generation. The agent-based approach uses reward driven software agents that are tasked with achieving the verification objectives, e.g. reaching coverage targets. A set of software agents can then be directed to interact, coordinating their behaviour in response to the system's observed actions in order to increase the likelihood of finding the edge case required to reach coverage targets more *efficiently*.

The agent-based technique creates two new entries in that taxonomy, a new Paradigm under Model Specification, *Agent-based*, and a new Technology under Test Generation, *Agency*, which includes reactive reasoning, causality and strategic planning between multiple agents in the test environment.

Machine Learning for Test Generation

The use of agent-based test generation brings up the debate of how such agents are instructed or trained to find their verification goals? A traditional MAS (Multi-Agent Systems) approach would set rules for each agent to follow based around the BDI (Belief Desire Intention) framework[9]. This approach requires each agent to perceive their environment, itself and other agents which forms a set of beliefs. Agents can select plans based around their intentions which are guided by their belief set to reach their goals (desires). The BDI approach to achieving agent goals can be very effective[10], but generating sufficient rulesets for large complex systems could be onerous and not conducive to an automated pipeline required for CyRes.

An approach to this issue may be to allow the agents to learn part of the ruleset by trial and error. This combines an element of randomness during a period where the agent learns which actions that result in favourable outcomes. The approach of Q-learning[11] is model-free and needs limited knowledge of the domain or the rewards beforehand, but relies on the agents to learn the optimal action policy based on reinforcement learning. This approach trades flexibility with a period of learning. Agents are assigned goals which are transferrable to any domain and therefore *robust but* must spend an amount of time to find an optimal control policy.

One of the keys to developing *robust* agents that can learn generalisable policies suitable for new and unseen domains is to expose the agents to a diverse set of training domains through randomisation of the parameters in the simulation environment. This is an extension to the original agent-based test generation approach[12] where the agents themselves are initialised in different

[8] M. Utting, A. Pretschner, and B. Legeard, "A taxonomy of model-based testing approaches," Software Testing, Verification and Reliability, vol. 22, no. 5, pp. 297– 312, 2012.

[9] Araiza-Illan, D., Pipe, A. G., & Eder, K. (2016). Intelligent agent-based stimulation for testing robotic software in human-robot interactions. ACM International Conference Proceeding Series, 9–16. https://doi.org/10.1145/3022099.3022101

[10] G. Chance, A. Ghobrial, S. Lemaignan, T. Pipe and K. Eder, "An Agency-Directed Approach to Test Generation for Simulation-based Autonomous Vehicle Verification," 2020 IEEE International Conference on Artificial Intelligence Testing (AITest), Oxford, UK, 2020, pp. 31-38, doi: 10.1109/AITEST49225.2020.0001

[11] Watkins, C. J. C. H., & Dayan, P. (1992). Q-Learning (Vol. 8).

[12] G. Chance, A. Ghobrial, S. Lemaignan, T. Pipe and K. Eder, "An Agency-Directed Approach to Test Generation for Simulation-based Autonomous Vehicle Verification," 2020 IEEE International Conference on Artificial Intelligence Testing (AITest), Oxford, UK, 2020, pp. 31-38, doi: 10.1109/AITEST49225.2020.00012.

locations for each training episode yet the static environment itself remained unchanged. However, the objective in both approaches is the same, diversify the experience of the agents to generate more generalisable action policies.

Some researchers suggest using a second learning algorithm to generate the randomisation of the training environment that uses a min-max adversary that seeks to minimise the performance of the verification agent providing it an opportunity to find and learn from the weaknesses in its policy. This can sometimes lead to unsolvable scenarios where the adversary always succeeds by creating domains in which the verification agent can only lose. New research suggests improvements to this by limiting the adversary to generate domains that are solvable in a method termed Protagonist Antagonist Induced Regret Environment Design (PAIRED)[13].

Reinforcement learning has been shown to solve large and complex problems in a timely fashion. An example is the success of the AlphaGo program[14], which used 1202 CPUs distributed across several servers to analyse 100's of possible moves within the time restriction of the game, typically 5s per move during the development cycle of the program. This demonstrates that the reinforcement learning approach can work at a large scale on a complex problem and, with the appropriate computational infrastructure, it returns results in a timeframe suitable for it to be used at runtime as part of the CyRes method.

Generally, machine learning tends to be limited to the training data available to it and as such can fail to generalise an agent's actions to unseen situations. This can be, and is, successfully averted by the use of simulation where new training data can be generated until confidence in the action policy meets the requirements for the agent's objectives.

Test Case Prioritisation & Labelling

Following test case generation there may be merit in arranging the execution order of the tests to promote those more pertinent to safety related issues for the design case in question. For example, if a modification was performed to the design in 'sub-system A' the test prioritisation could be ranked to ensure safety and functionally critical tests related to that subsystem are executed first.

This approach aims to uncover flaws in the modified system early on, rather than using a fixed order of tests or using random test selection. By detecting a design's functional flaws or safety concerns early, subsequent test cases need not be executed, potentially saving significant computational resources. Designs that have been flagged as flawed or unsafe due to assertion violations may be immediately discarded and priority can then be given to another design from the candidate cases.

In order for the prioritisation to operate successfully at runtime, test case labelling must be completed prior to operational readiness. Test cases must be labelled to indicate what aspect of the verification process they attend, what coverage they have achieved. This allows a set of test cases to be called upon that can assess specific functionality or safety requirements. This collection of labelled tests will be compiled at the design stage of CyRes and will optimise runtime performance.

[13] Dennis, Michael, et al. "Emergent Complexity and Zero-shot Transfer via Unsupervised Environment Design." arXiv preprint arXiv:2012.02096 (2020).

[14] Silver, D., Huang, A., Maddison, C. et al. Mastering the game of Go with deep neural networks and tree search. Nature 529, 484–489 (2016). https://doi.org/10.1038/nature16961

Online Verification for CyRes Advance

Building on the success of previous research at TSL[15] and the future trends in building generalisable agents that can succeed in unseen domains[16], we propose the following process to the CyRes method. In Figure 23 we illustrate the fundamental principle, which follows the test case for a braking system and using online simulation to provide rapid verification assessment.

The verification requirements were defined and written prior to operation and potentially remain invariant throughout the assessment lifecycle barring any significant changes to the expected behaviour of the system. These requirements must be accessible and parseable to the online simulation and guide the test generation method towards achieving the goals of the verification, be that discovery of pertinent bugs or any prescribed coverage that is needed.

An automated testbench manager had access to the verification requirements and used these to make decisions on the types of test cases that needed to be executed. The testbench manager invoked test cases from a set of test generation method classes. These classes included simple random techniques which can be parametrically searched. Another option was to use existing or known test cases from a test case library which may be pertinent if sub-systems of the DUV are identical and simply require a confirmation that expected functionality and safety also remains unchanged. An ML (Machine Learning) and AI test generation methods class would also be available as part of the test generation methods group. It was invoked to provide intelligent, reactive test cases.

Within the ML & AI Class the testbench manager used the verification requirements to target specific verification goals, invoking tests that drive the simulation towards particular coverage. Previous work (see footnote 11) has shown effectiveness in giving agency to dynamic elements within the simulation but current research trends also show that an improvement in effectiveness can be gained through use of an adversarial domain agent responsible for the generation of the static environment which can further enhance the effectiveness of the test cases.

Invoked test cases generated the static and dynamic elements required for the simulation to run. A design case was chosen from the set of candidate cases and tested in the simulation. The simulation job being sent to a distributed cluster to execute. It is possible that groups of jobs can be sent simultaneously and executed in parallel for accelerated progress.

Upon completion of the executed job or batch of jobs, assertion testing on the simulation log files can be performed, either locally or remotely, to check for assertion violations. These indicate a failure with respect to the verification requirements. During testing, each activated assertion, whether passed or failed by the tests, contributes to test coverage. The testbench manager repeatedly invokes tests to collect further coverage until there is satisfactory coverage across all verification requirements. Initial performance evaluation of this approach suggests that, instead of post-processing after the simulation has terminated, assertion checking can reasonably be performed on-the-fly, i.e. during simulation. This has the benefit that simulation could in principle be terminated as soon as the first assertion violation for a given design modification has been detected, saving valuable simulation resources and time.

[15] G. Chance, A. Ghobrial, S. Lemaignan, T. Pipe and K. Eder, "An Agency-Directed Approach to Test Generation for Simulation-based Autonomous Vehicle Verification," 2020 IEEE International Conference on Artificial Intelligence Testing (AITest), Oxford, UK, 2020, pp. 31-38, doi: 10.1109/AITEST49225.2020.00012.
[16] Dennis, Michael, et al. "Emergent Complexity and Zero-shot Transfer via Unsupervised Environment Design." arXiv preprint arXiv:2012.02096 (2020).

If the chosen design violates one or more assertions, then the testbench manager may invoke, resources permitting, further tests within the same area to determine if the violation is a single event or a trend. These designs can be considered failures as they failed to meet the requirements for verification. At this point the design can be discarded from the set of candidates or indicated in a report log that the design was excluded for failing certain tests.

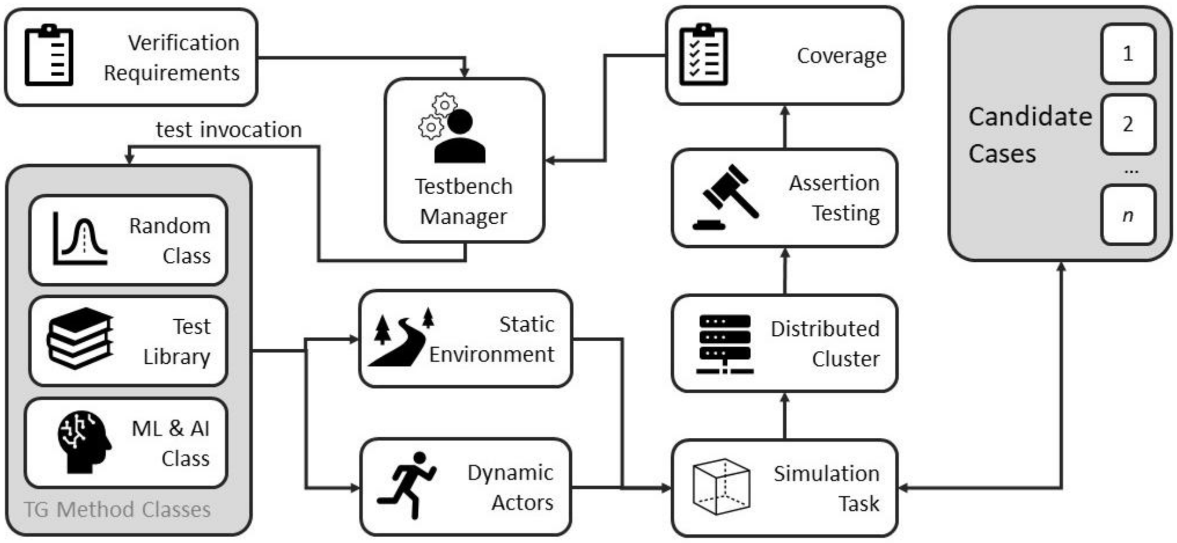


Figure 23 Use of simulation for online verification in the CyRes framework.

Distributed Ledger

When operational software is changed a change log is usually included, detailing all the updates and changes that were made. For the CyRes methodology a similar process is proposed by using a distributed ledger. For the online verification process this is a useful system to ensure secure record keeping of what design changes were made to the system and the verification tasks that were completed.

The ledger may collect details such as the tests performed on each design and a log of the key outcomes and whether any assertion violations were found. It could further log the state of the test system which we have shown to be important in the reliability of certain simulation code. Details such as the build and configuration of the remote simulation server, simulation code version and even runtime specifics such as resource utilisation, scheduling policy and process priority have all been shown to impact variance between otherwise identical simulation tasks[17].

The ledger may also serve as an executive level account of the proposed design and those which have failed to meet the verification requirements, and which have been put into operation.

Deterministic Simulation

The use of simulation for testing and verification has become ubiquitous throughout research and commercially focussed organisations. Simulation can be seen as the less costly and complementary method to physical testing, also allowing safety-critical tests to be performed without consequences to the safety of trial participants.

[17] G. Chance, A. Ghobrial, S. Lemaignan, T. Pipe and K. Eder, "On Determinism of Game Engines used for Simulation-based Autonomous Vehicle Verification", arxiv preprint

Increasingly, the autonomous vehicle community are adopting game engines as simulation platforms to support the development and testing of vehicle control software. CARLA[18], for instance, is an open-source simulator for autonomous driving that is implemented in the Unreal Engine[19], a real-time 3D creation environment for the gaming and film industry as well as other creative sectors.

State-of-the-art game engines provide a convenient option for simulation-based testing. They offer sufficient realism in the physical domain combined with realistic rendering of scenes, potentially suitable for perception stack testing and visual inspection of accidents or near misses. Furthermore, they are easy to set up and run compared to on-road testing and are simple to control and observe, both with respect to the environment the AV operates in as well as the temporal development of actors[20]. Compared to the vehicle dynamics simulators and traffic-level simulators used by manufacturers[21], game engines offer a simulation solution that meets many of the requirements for the development and functional safety testing of AVs in simulation. However, while game engines are designed primarily for performance to achieve a good user experience, the requirements for AV verification go beyond that and include determinism.

TSL has shown that game engines are not deterministic and under certain conditions fail to meet the tolerances required for verification. In Figure 24 the results of a case study are presented for a verification environment that uses CARLA. The maximum variance in the simulation output (y axis) is plotted for various simulation tasks (tests 1-6) that cover different situations involving vehicles and pedestrians in an autonomous driving scenario. The required tolerance of 1cm for this example is indicated in Figure 24 with a dashed 1cm line. Additionally, the level of system resource utilisation was artificially changed to imitate a simulation server under a high load (x axis).

The study found, for the specific system investigated, that scenarios that involved vehicle collisions (tests 2 & 4) resulted in a variance consistently over the maximum specified for the verification outcome to be reliable, irrespective of whether the system was under high computational stress or not. Intuitively, the calculations required to simulate the physical processes associated with vehicle collisions (as in tests 2 & 4) can reasonably be expected to be significantly more complex, causing increased computational load, than those where no vehicle collisions occur (e.g. tests 1 & 3). In fact, the scenarios with no vehicle collisions (tests 1 & 3) show low simulation variance under light computational load. However, artificially increasing the computational load for these scenarios appears to result in increased simulation variance, with very high computational stress (>=75% resource utilisation) leading to variance above tolerance, similar to that observed for the scenarios with collisions.

[18] CARLA: Open source simulator for autonomous driving research. http://carla.org/. Accessed: 2020-011-13.

[19] Unreal Engine 4. https://www.unrealengine.com/. Accessed: 2020-011-13.

[20] S. Ulbrich, T. Menzel, A. Reschka, F. Schuldt, and M. Maurer, “Defining and Substantiating the Terms Scene, Situation, and Scenario for Automated Driving,” IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC, vol. 2015-Oct, pp. 982–988, 2015.

[21] Z. Saigol and A. Peters, “Verifying automated driving systems in simulation framework and challenges,” 25th ITS World Congress, Copenhagen, 2018.

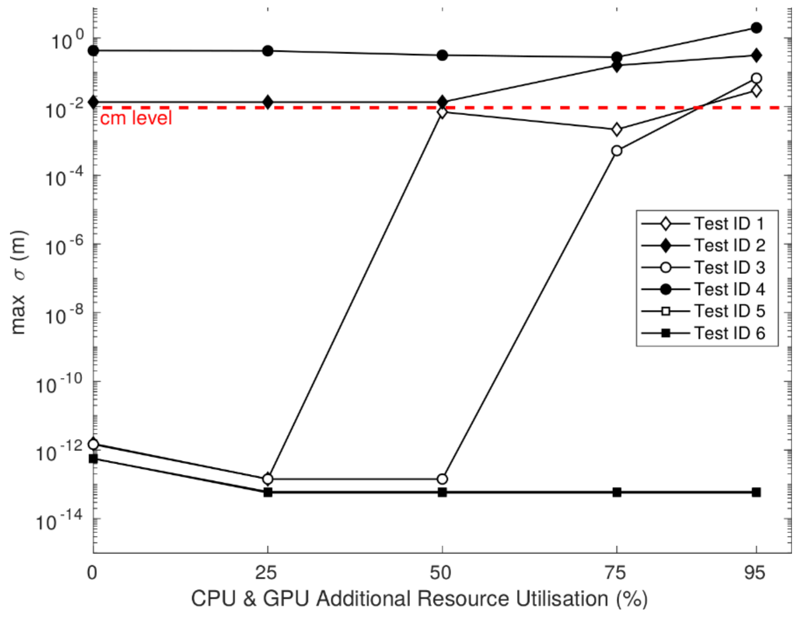


Figure 24 Simulation variance for various verification tasks and resource utilisation levels.

The high computational load to simulate physical processes can reasonably be linked to an increase in thread activity and therefore CPU scheduling. The research postulated that a change in execution order, e.g. caused by CPU scheduling, could be a key source of simulation variance, because when certain calculations are processed in different orders, they can produce different results, e.g. due to floating point arithmetic not being associative.

The work proposes a general process that can be followed to determine whether a simulation environment is deterministic and, if not, then what can be done to understand the potential sources of variance and how the system should be monitored at runtime to ensure compliance to an agreed upon tolerance, Figure 25. In any operational system for CyRes, it would be important to track aspects of the simulator in the internal and external settings shown in items 2 & 3 such as; process priority for the simulation and any other processes running concurrently, CPU and GPU utilisation levels, memory configuration and any specified memory placement if using clustered or virtualised machines, specific configurations and version of the simulation code and a full description of the scenario using e.g. the open-source format openScenario[22].

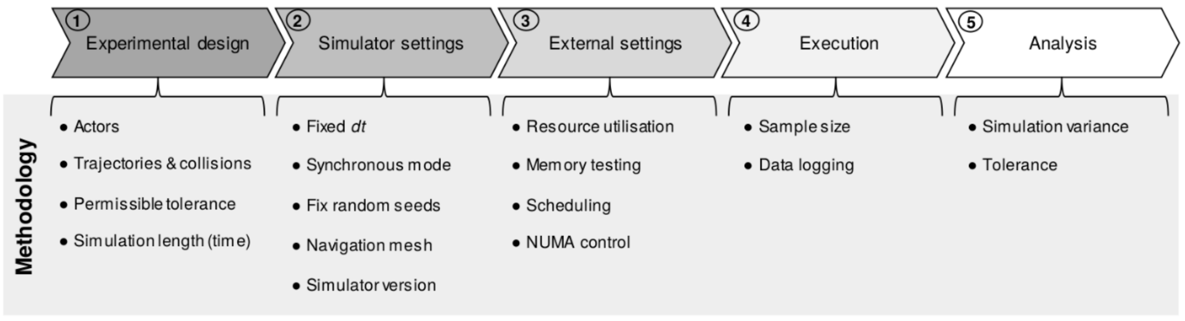


Figure 25 Methodology devised for determining simulation variance.

[22] https://www.asam.net/standards/detail/openscenario/

Automated/Online Assertion Checking

We present a demonstration of the assertion testing process that can be applied to any system with the appropriate monitoring. In the example below a driving scenario is considered.

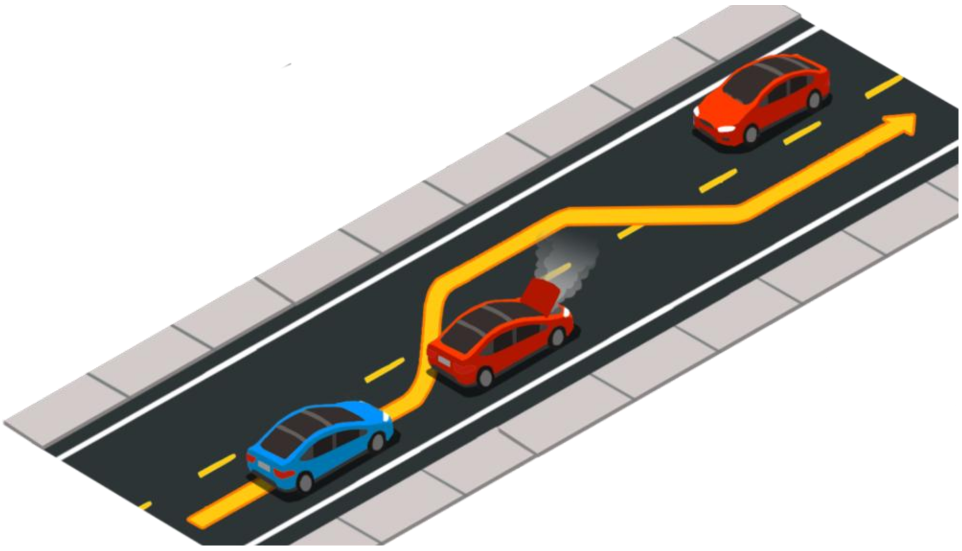


Figure 26 An overtaking scenario from https://carlachallenge.org/challenge/nhtsa/.

The simulation environment can be a full 3D rendered environment that is rich in details suitable for testing and verification of, for example, perception stacks, where details such as broken road lines or cloud cover could result in different interpretation of a scene from the perspective of the control system. While this high level of detail is needed for testing, only an abstracted version of events is required for many of the online verification tasks, especially those that can be tested logically against a fixed ground truth.

For the overtaking scenario, the driver or autonomous control system must make a judgement prior to and during overtaking that complies with a 'ground truth' that we wish to observe. The rules of this ground truth may be formulated in many ways and are typically based on the rules and standards that can be found in the highway code, driving laws or social norms.

In the following example we use the rules from entry 162 of the UK Highway Code to illustrate the creation of an assertion in a format that can be tested in simulation. Rule 162 states:

Before overtaking you should make sure

- the road is sufficiently clear ahead,
- road users are not beginning to overtake you,
- there is a suitable gap in front of the road user you plan to overtake.

Formalisation of Assertions

The next step is to turn these human interpretable rules, regulations and laws into a language that can be computationally parsed. These machine-readable rules are called a assertions. In general, we distinguish invariants, which are properties that must be satisfied throughout a scenario or part of a scenario, e.g. observing a given speed limit or maintaining a safe distance to other road users, as well as pre- and post-conditions, and potentially variant properties. To cover a highway rule completely, a combination of logic properties will normally be required.

Data obtained during the simulation can then be used to check the assertions that are relevant for the given scenario. This can be achieved using assertion monitors, which compute the truth value for logical expressions from the simulation logs. Figure 27 illustrates the steps taken to generate an assertion checking algorithm for the overtaking scenario. This includes the parameters and metrics that need to be monitored at runtime. For overtaking this requires at least knowledge of the position of the vehicle and the lead vehicle to be overtaken, the details of the road and road markings including where the legal centreline of the road is, the speed of each actor and other dynamic data such as the braking distance of each vehicle.

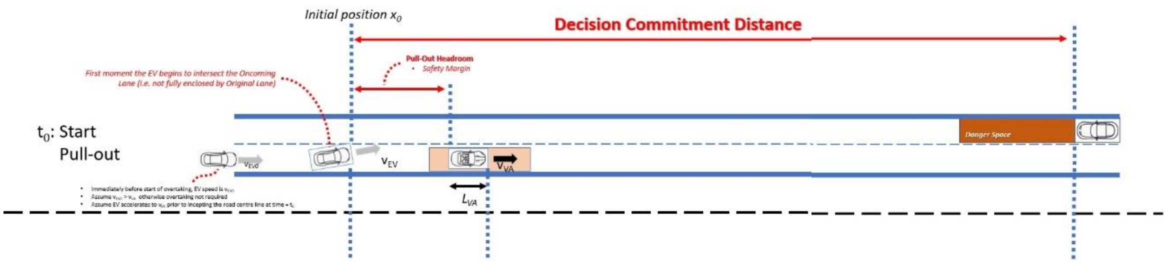

Initial position x_0
Decision Commitment Distance
Pull-Out Headroom
• Safety Margin
First moment the EV begins to intersect the Oncoming Lane (i.e. not fully enclosed by Original Lane)
t_0: Start Pull-out
v_{EV0}
v_{EV}
v_{VA}
L_{VA}
Danger Space
• Immediately before start of overtaking, EV speed is v_{EV0}
• Assume $v_{EV0} > v_{VA}$ otherwise overtaking not required
• Assume EV accelerates to v_{EV} prior to incepting the road centre line at time = t_0

Figure 27 Analysis of an overtaking scenario to determine an assertion checking routine.

Assertion Checking

The process of checking the assertion must only use metrics that can be monitored at runtime, or at least calculated within reasonable time from raw simulation log data, e.g. if speed is not monitored directly then it can be calculated from position and time data.

For assertion checking the level of detail need not be at the level of fidelity of the simulator but just detailed enough to capture the metrics required for the purposes of assertion checking. As such, lightweight logfiles are usually sufficient and can be checked offline or run online alongside the simulation itself. These logs will be useful for any regulatory or legal requirements of the CyRes process and due to being relatively light data, could be stored in the distributed ledger.

Figure 28 shows a layout of the database structure that can be used for assertion checking. The simulation data is written to the *raw_data* table under the main *sim_log*. The log also contains details about any relevant contextual information under the *environment* header, which includes road map information for the overtaking example including lane delineation and the position and type of road markings. This raw data will also capture status conditions of actors which may include data such as indicating status or traffic light signal colour.

The top left of Figure 28 shows the storage of the assertions in a database file which may be written in human readable form and is then converted to a form readable by the database. In this example, SQL is appropriate for the geospatial relational queries required to check assertions.

An important part of the assertion checking process requires *dynamic* information which may be further enriched by combining this with functional environmental data (functional_static) which might include occluding objects (foliage) or areas of varied illumination. This level of detail will allow queries to include whether actors are visible or have crossed road lines and be able to project braking distances based on speed, maximum braking deceleration and heading angle. Dynamic queries are especially useful to enable testing complex assertions during online verification.

A precondition monitor (lower right in Figure 28) will only activate the assertion check if the precondition is met. For example, violation of a red traffic light can only be tested if there is a red light present when the vehicle approaches the signal controlled area. This approach saves resources and superfluous log entries. If an assertion is checked, the SQL query is executed and the results are written to the *assertion_results* table. Finally, a database of actors can be called upon to draw down information or characteristics about the actors within the simulation which may include information that will assist with the assertion checking such as actor size and shape, acceleration and deceleration limits, centre of gravity and any other actor-specific data.

In conclusion, a database structure like the one shown in Figure 28 would be suitable for assertion testing in the CyRes framework. The database proposal would work on a large scale and operates on libraries that have fast access and execution times, being commercially used for large scale operation such as Google maps.

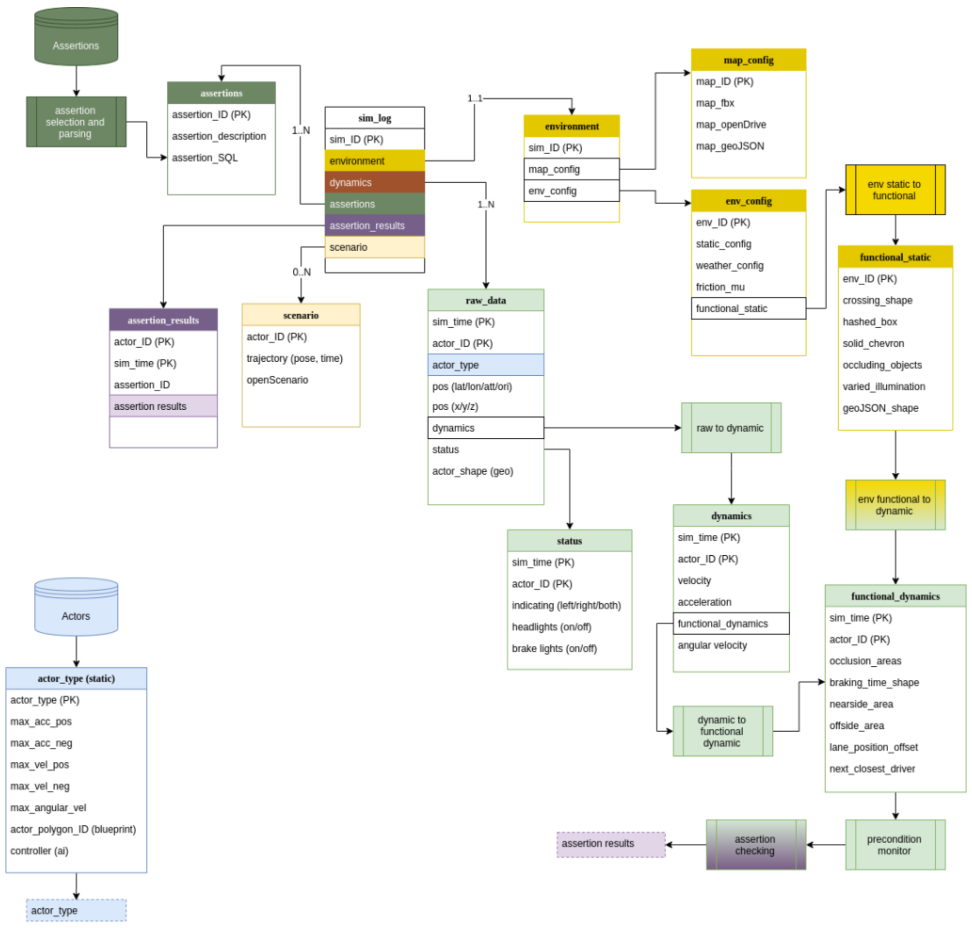


Figure 28 A database structure suitable for assertion checking of an autonomous system.

Appendix F: The Stability Analysis Tool

Conceptual Design

In principle, the stability analysis framework is developed through two phases, offline design and online operation, as depicted in **Error! Reference source not found.**.

In the offline design phase, Lyapunov stability theory is employed to establish a stability rule working within a set of constraints to assess the control system stability. The system is considered as “stable” if all of the rule and constraints are satisfied. The system is considered as “unstable” if either the rule or any of the constraints are unsatisfied. In addition, a system observer is

In the online operation phase, at each step (based on the assumptions made during the problem definition):

- If any of the stability constraints or the stability rule derived in the offline phase is not satisfied, the system is unstable and an alarm is triggered for the system to make an immediate decision
- DoS attack is triggered firstly based on the system feedback delay (time stamp of event driven approach).
- Deception attack is triggered based on the difference between the actual system performance measurement and the observer output (performance divergence).
- Two counters are used to count the numbers of consecutive DoS and Deception attack detections, respectively
    - If any/both counter value is/are above the pre-defined threshold(s) (called a timeframe window)
        - DoS or/and Deception attack
        - If the system is still identified as stable, only an alarm is triggered but it may allow the vehicle to continue operation. In addition, AI-based machine learning module (if having) is enabled to update the database to accept this minor attack in the future
        - If the system is unstable, a mitigation plan is required, for example, to send warning to the user, to reconfigure the control system within some thresholds or to stop the vehicle immediately.
    - Else, the system can continue its operation as long as it is stable

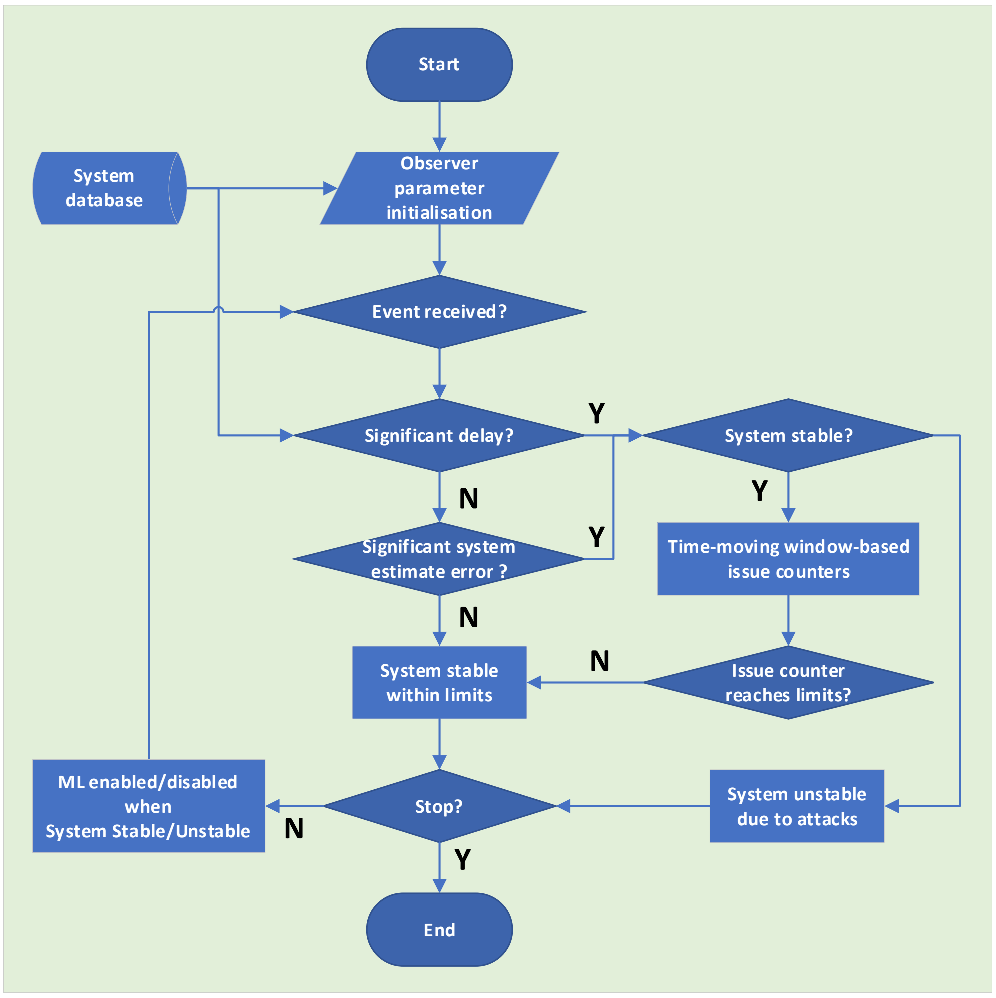

Start
System database
Observer parameter initialisation
Event received?
Significant delay?
Y
System stable?
N
Y
Y
Significant system estimate error ?
Time-moving window-based issue counters
N
N
System stable within limits
Issue counter reaches limits?
ML enabled/disabled when System Stable/Unstable
Stop?
System unstable due to attacks
N
Y
End

Figure 29 Proposed design framework for stability analysis tool

Lyapunov Stability Theory

Design Principles

System model

Without loss of generality, a dynamic model of a control system in CAVs can be represented as

$$\begin{cases} \dot{x}(t) = Ax(t) + Bu(t) + Dw(t) \\ y = Cx(t) \end{cases} \tag{1}$$

where $x \in R^x, u \in R^u, y \in R^y$ and $w \in R^w$ denote the state vector, control input, measurement from sensors and disturbances, respectively; *A*, *B*, *C* and *D* are matrices with appropriate dimensions (either time-variant or time-invariant).

Design a generic control law for the system in the form of:

$$u(t) = KCx(t) \tag{2}$$

Modelling a system controlled under an imperfect network

Without loss of generality, a networked control system (NCS) consists of actuators and sensors which are time-driven with a sampling rate of *h* and a controller which is zero-order hold (ZOH) and event-driven (e.g. via CAN). Considering both network communication issues (delays, package dropout and disordering), a CAV control system in **Error! Reference source not found.** can be r epresented as

$$x(t) = Ax(t) + BKCx\big(t - \theta(t)\big) + Dw(t) \tag{3}$$

where $\theta(t)$ is a piecewise linear function satisfying

$$\tau_{t_k} \leq \theta(t) \equiv t - t_k h \leq (t_{k+1} - t_k)\, h + \tau_{t_{k+1}} \tag{4}$$

To minimise the network traffic, a threshold can be added in to the event-driven communication scheme as (using a scalar parameter σ and a positive matrix Φ)

$$t_{k+1}h = t_k h + min\{ph | e^T(i_p h)\Phi e(i_p h) > \sigma x^T(t_k h)\Phi x(t_k h)\} \tag{5}$$

where $i_p = (t_k + p)h;\, e(i_p h) = Cx(i_p h) - Cx(t_k h)$.

The condition **Error! Reference source not found.** means that the even-driven communication is o nly activated once there is a sufficient amount of change at the measurement site.

Following **Error! Reference source not found.**, define a new piecewise linear function:

$$\tau_{t_k} \leq \eta(t) \equiv t - i_p h \leq (t_{k+1} - t_k)\, h + \tau_{t_{k+1}} \tag{6}$$

A generic NCS in CAVs can be modelled as

$$\begin{cases} \dot{x}(t) = Ax(t) + BKCx\big(t - \eta(t)\big) + BKCe\big(i_p h\big) + Dw(t) \\ \qquad\qquad\qquad\qquad y = Cx(t) \end{cases} \tag{7}$$

Stability analysis of formulated problem

Without loss of generality, system **Error! Reference source not found.** always can be represented in a nother form:

$$\dot{x}(t) = F(t, x(t)) + G(t, x\big(t - \eta(t)\big) \tag{8}$$

Based on the system presentation **Error! Reference source not found.** and if the delay is bounded, o ne of the most well-known techniques to ensure the resilient control is to define a Lyapunov-Krasovskii function $V\big(x(t)\big)$ such that $V\big(x(t)\big) > \varepsilon |x(t)|^2, \varepsilon > 0$ and along the trajectories of **Error! R eference source not found.**:

$$\dot{V}(t) = -x^T(t)Qx(t) \tag{9}$$

The Lyapunov-Krasovskii function is then designed in such a way that the system is guaranteed to be asymptotically stable with a control gain *K*:

$$V(P) = P^T(0)U(0)P(0) + 2P^T(0)\int_{-\eta(t)}^{0} U^T(\eta(t) + \gamma)GP(\gamma)\, d\gamma + \iint\limits_{-\eta(t)}^{0} P^T(\gamma_2)G^T U(\gamma_2 - \gamma_1)GP(\gamma_1)\, d\gamma_1\, d\gamma_2 \tag{10}$$

where

$$U(\gamma) = \int\limits_0^\infty K^T(t)\, QK(t + \gamma)dt \tag{11}$$

By differentiating $V\big(x(t)\big)$ along the trajectory of **Error! Reference source not found.**, the control g ain *K* can be found to ensure that $\dot{V}(t) < 0$.

Scalability and Adaptability

Conceptual Design

The control scheme proposed in the previous sections is developed based on the system's mathematical representation **Error! Reference source not found.** and/or **Error! Reference source no t found.**. Regarding practical applications to different systems in CAVs, especially with impacts of engineering differences (e.g. environment, network bandwidth, and system ageing), there are always unpredictable uncertainties and nonlinearities. This will lead to the mismatch between the actual system performance and its estimated performance using **Error! Reference source not found.** a nd/or **Error! Reference source not found.**.

The following approach can be utilised to improve the scalability of the tool:

- First, based on the design knowledge, each system can be modelled in a state space form
- Second, all unknown nonlinearities and uncertainties (deducted from the bounded delays formulated in **Error! Reference source not found.**), can be represented by one o r several lumped – unknown nonlinear function(s) – representing by time-variant matrices as denoted in **Error! Reference source not found.**.
- Third, any unknown nonlinear function can be modelled by using AI techniques and Lyapunov-based machine learning (ML) technique. Acceptable instantaneous divergence between the actual system response and its estimated value will be employed to support ML.

Furthermore, the following approach can be utilised to improve the adaptability of the tool:

- If and either DoS or Deception attack (or both of them) is detected while the system is still stable, an impact analysis should be carried out to evaluate the potential of impact (risk evaluation) if continuing to operate the vehicle under this attack.
- If the risk score is very low following the BS ISO/SAE 21434:2021 standard, the AI-based ML mechanism can be designed to update the database to accept this type of attack in the future. It means the system will adapt to the new working condition by some potential routes as follows:
  - Refining the stability constraints
  - Regulating the timeframe window
  - Regulating the threshold of time delay (indicating a potential DoS attack) and/or the threshold of performance divergence (indicating a potential of Deception attack)

Motivation

As we introduce diversity into the CAV systems, and consider vulnerability analysis of the implemented systems, there is a need to consider the stability of the systems. One example of where diversity may be introduced is in the Networked Control System (NCS). Potential cyber attacks on the NCS include Denial-of-Service (DoS), deception attack and attacks to involving system time-delays and/or package disorders.

There is a need to study how stability analysis theories, such as Lyapunov, can be utilised to distinguish the impact of cyber attacks and natural system nonlinearities - uncertainties (including issues associated with imperfect network communication) in real-time.

Technical approach

First, we define the set of systems that are proposed through the diversity-by-design strategy. We next define the potential attack points at both the local network within a vehicle and the global network between vehicles.

Third, we represent the problem through a defined mathematical model. Attack types such as DoS, time-delays and package disorders can be represented in a form of modified NCS model(s) in which thresholds could be used to classify between: natural time-delays, natural package losses, attacked time-delays and attacked package losses. Meanwhile deception attacks through vehicle sensor information could be modelled by employing either nonlinear functions (if sub-system/component models available) or unknown nonlinear functions (using AI, like fuzzy or neural network). Those functions can be bounded by vehicle sub-system/component specifications and power – working ranges. Impact of deception attacks on sensor information can be then represented by a probability distribution function staying within the nonlinear functions.

Finally, Lyapunov theory will be deployed to ensure the stability of NCS(s). As a calibration tool, the proposed NCS' outputs can be used to compare with CAVs' decisions (made by their existing control units) to identify cyber attacks. The proposed NCS can also be employed directly in CAVs to support the vehicle decision making irrespective of attack conditions and levels.

Case study

Braking control system in CAVs can be selected as a case study. Generic block diagram of this braking control system with potential attack points can be depicted in Figure 30 below. In this figure, the proposed NCS tool is tagged as "Break Control Observer".

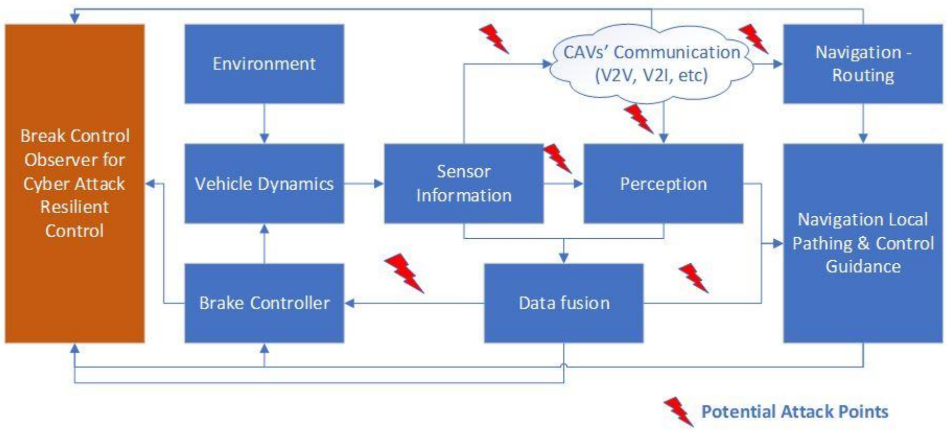
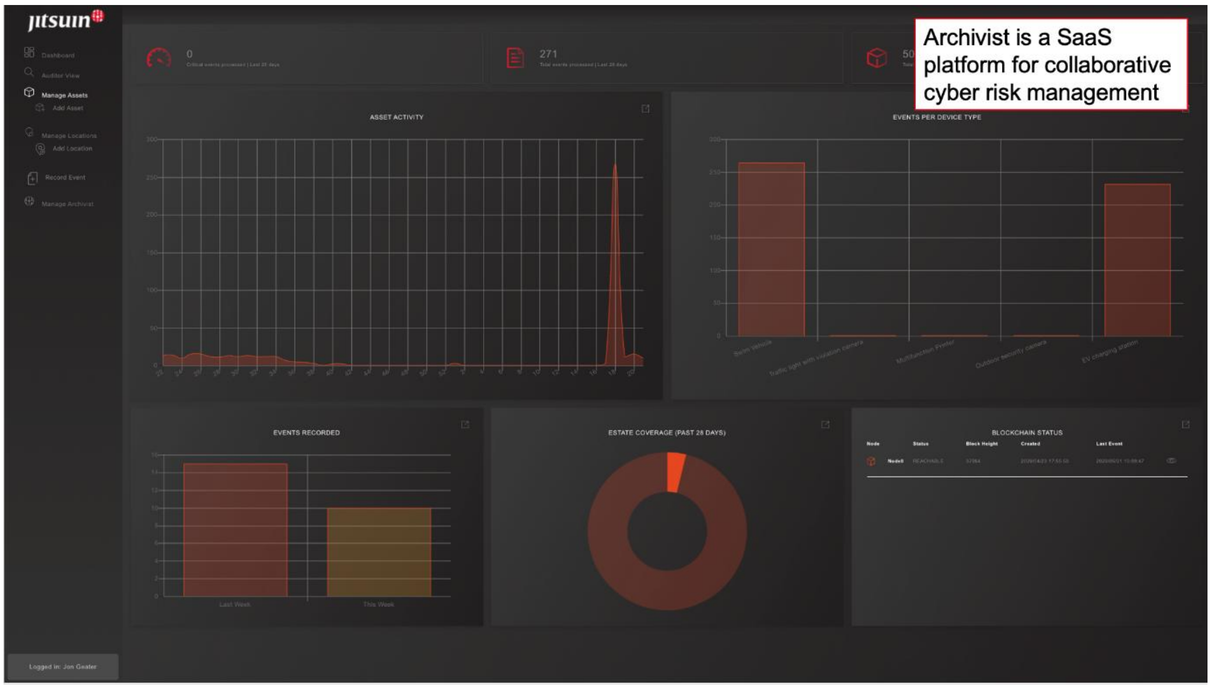
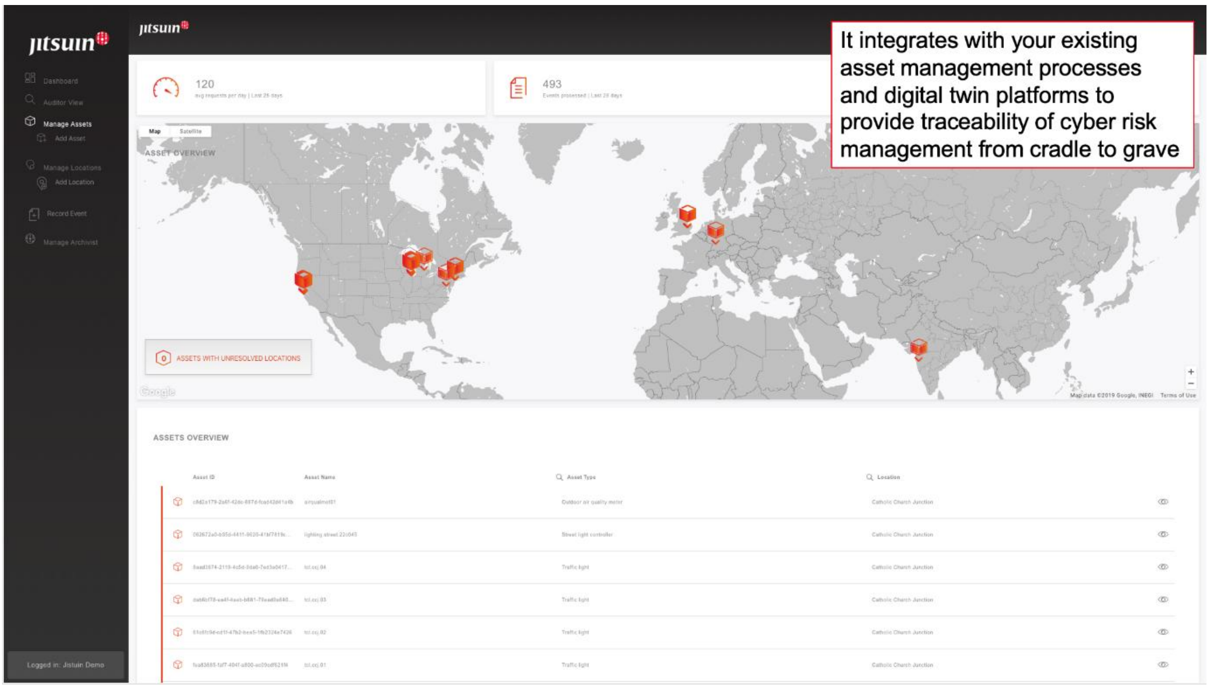

Break Control Observer for Cyber Attack Resilient Control
Environment
CAVs' Communication (V2V, V2I, etc)
Navigation - Routing
Vehicle Dynamics
Sensor Information
Perception
Navigation Local Pathing & Control Guidance
Brake Controller
Data fusion
Potential Attack Points

Figure 30: Generic block diagram of braking control system with potential attack points

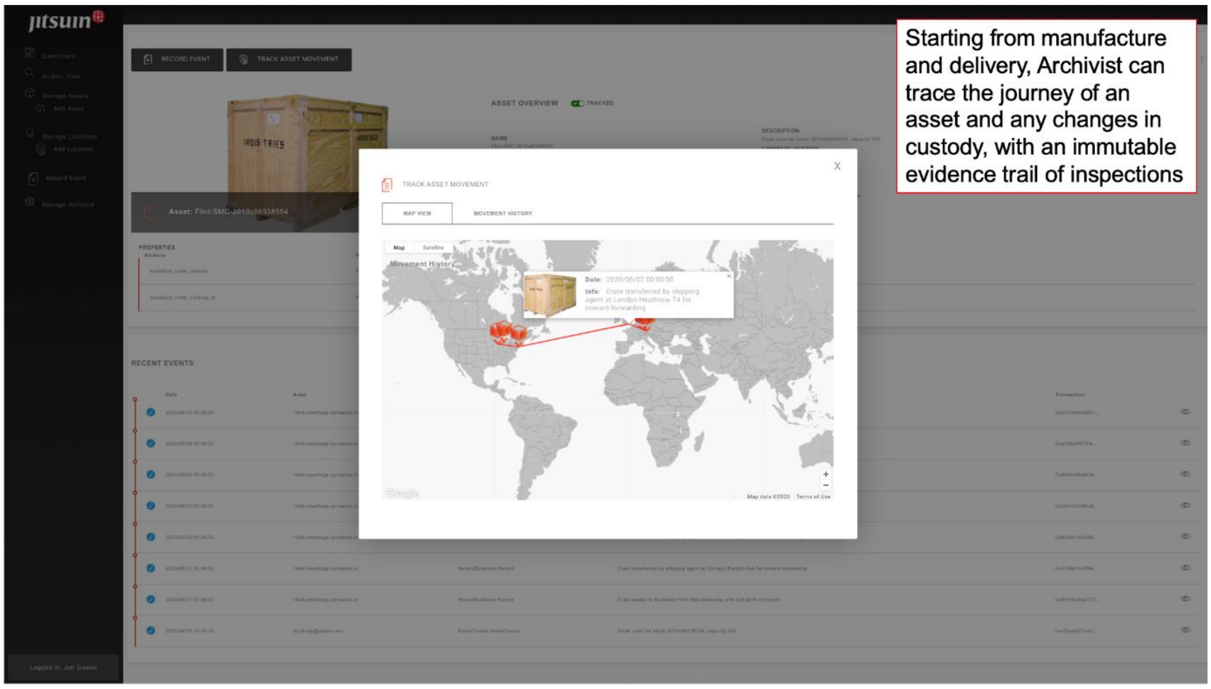
Appendix G: Illustrative Screenshots for RKVST



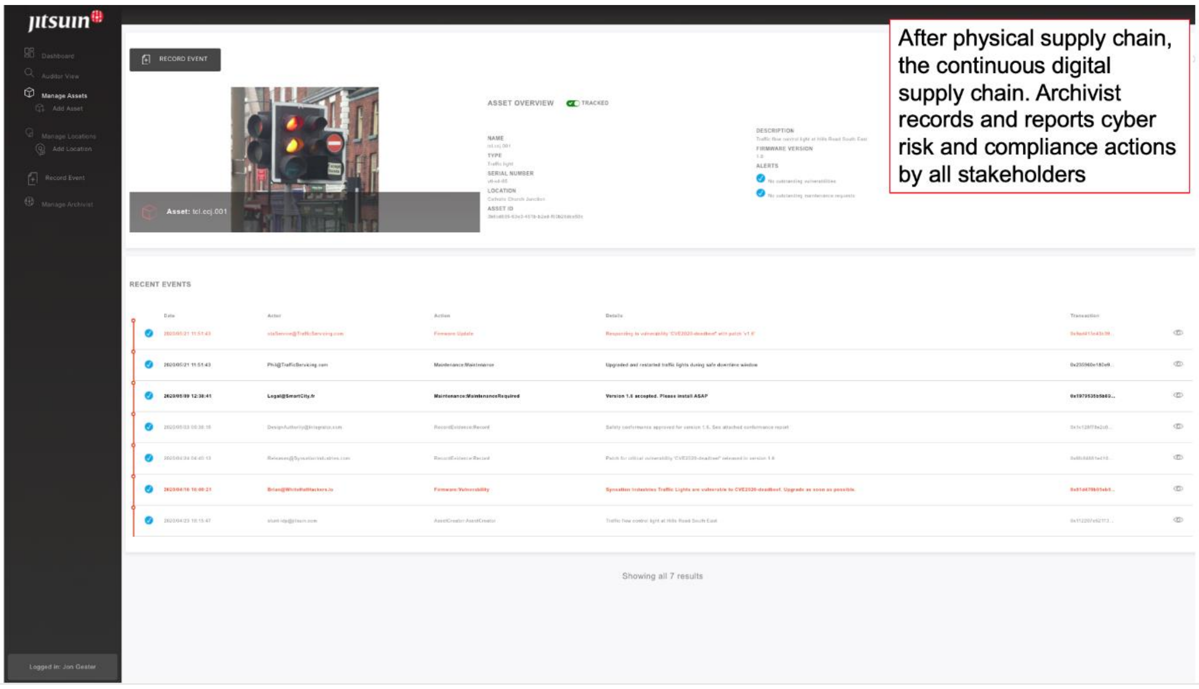
Archivist is a SaaS platform for collaborative cyber risk management



It integrates with your existing asset management processes and digital twin platforms to provide traceability of cyber risk management from cradle to grave



Starting from manufacture and delivery, Archivist can trace the journey of an asset and any changes in custody, with an immutable evidence trail of inspections

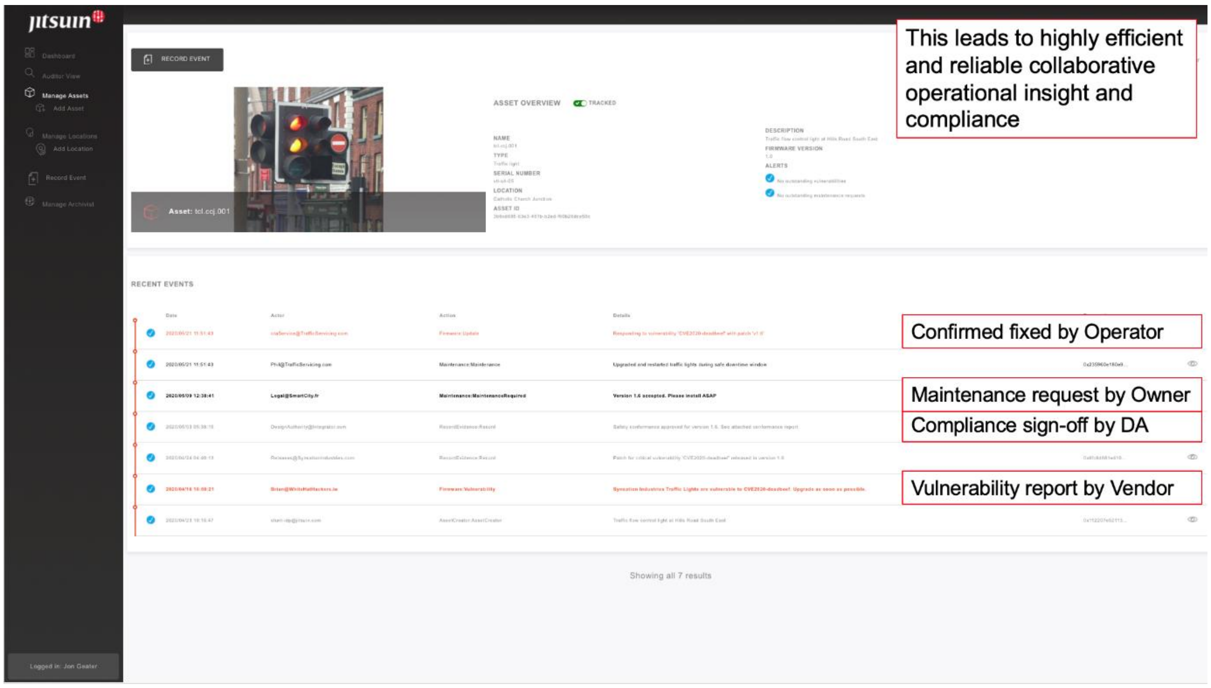


RECENT EVENTS

Date	Actor	Action	Details	Transaction
2020/05/21 11:51:43	[illegible]	Firmware Update	[illegible]	[illegible]
2020/05/21 11:51:43	Phil@TrafficServicing.com	Maintenance:Maintenance	Upgraded and restarted traffic lights during safe downtime window	[illegible]
2020/05/09 12:38:41	Legal@SmartCity.fr	Maintenance:MaintenanceRequired	Version 1.6 accepted. Please install ASAP	[illegible]
[illegible]	[illegible]	RecordEvidence:Record	[illegible]	[illegible]
[illegible]	[illegible]	RecordEvidence:Record	[illegible]	[illegible]
2020/04/16 16:00:21	[illegible]	Firmware:Vulnerability	[illegible]	[illegible]
[illegible]	[illegible]	AssetCreator:AssetCreator	Traffic flow control light at Hills Road South East	[illegible]

Showing all 7 results

After physical supply chain, the continuous digital supply chain. Archivist records and reports cyber risk and compliance actions by all stakeholders



ASSET OVERVIEW

NAME
tcl.ccj.001
TYPE
Traffic light
SERIAL NUMBER
vtl-a4-05
LOCATION
Catholic Church Junction
ASSET ID
3b6ed685-63e3-457b-b2ed-f60b26dce50c

DESCRIPTION
Traffic flow control light at Hills Road South East
FIRMWARE VERSION
1.0
ALERTS
✓ No outstanding vulnerabilities
✓ No outstanding maintenance requests

RECENT EVENTS

Date	Actor	Action	Details
2020/05/21 11:51:43	ctaService@TrafficServicing.com	Firmware:Update	Responding to vulnerability 'CVE2020-deadbeef' with patch 'v1.6'
2020/05/21 11:51:43	Phil@TrafficServicing.com	Maintenance:Maintenance	Upgraded and restarted traffic lights during safe downtime window
2020/05/09 12:38:41	**Legal@SmartCity.fr**	**Maintenance:MaintenanceRequired**	**Version 1.6 accepted. Please install ASAP**
2020/05/03 05:38:15	DesignAuthority@Integrator.com	RecordEvidence:Record	Safety conformance approved for version 1.6. See attached conformance report
2020/04/24 04:40:13	Releases@SynsationIndustries.com	RecordEvidence:Record	Patch for critical vulnerability 'CVE2020-deadbeef' released in version 1.6
2020/04/16 16:08:21	**Brian@WhiteHatHackers.io**	**Firmware:Vulnerability**	**Synsation Industries Traffic Lights are vulnerable to CVE2020-deadbeef. Upgrade as soon as possible.**
2020/04/23 10:16:47	stern-ida@jitsuin.com	AssetCreator:AssetCreator	Traffic flow control light at Hills Road South East

Showing all 7 results

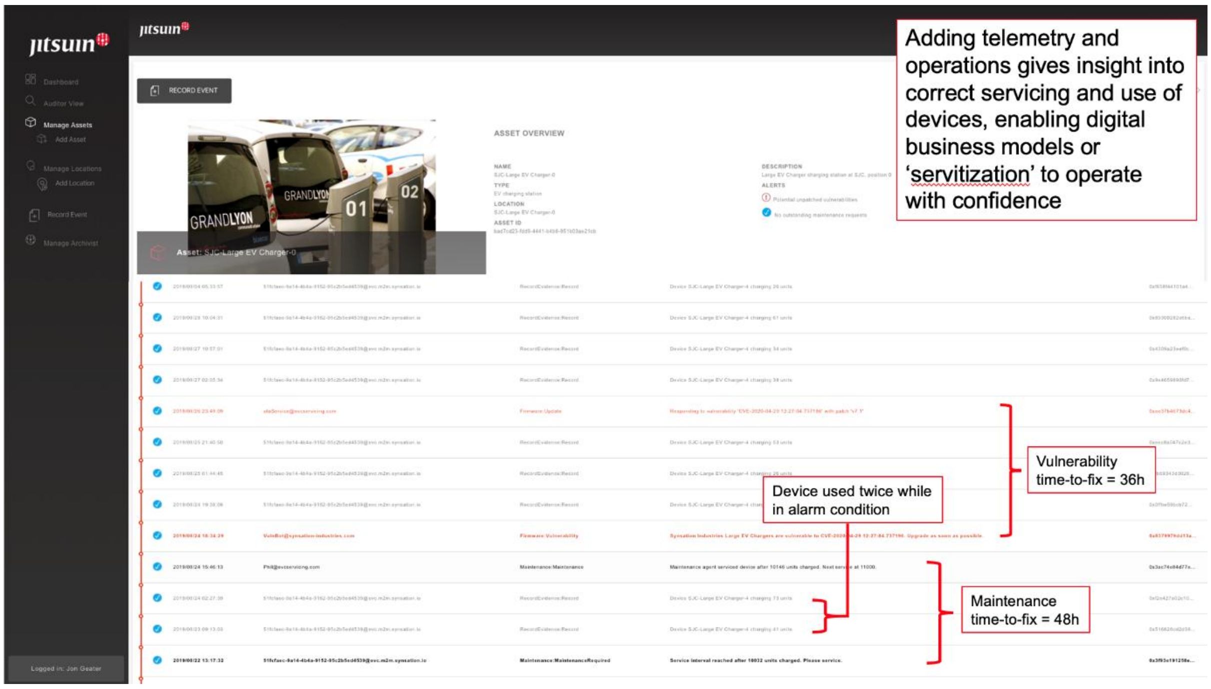
This leads to highly efficient and reliable collaborative operational insight and compliance

Confirmed fixed by Operator

Maintenance request by Owner

Compliance sign-off by DA

Vulnerability report by Vendor



ASSET OVERVIEW

NAME
SJC-Large EV Charger-0
TYPE
EV charging station
LOCATION
SJC-Large EV Charger-0
ASSET ID
bad7cd23-fdd5-4441-b4b6-951b03ae21cb

DESCRIPTION
Large EV Charger charging station at SJC, position 0
ALERTS
Potential unpatched vulnerabilities
✓ No outstanding maintenance requests

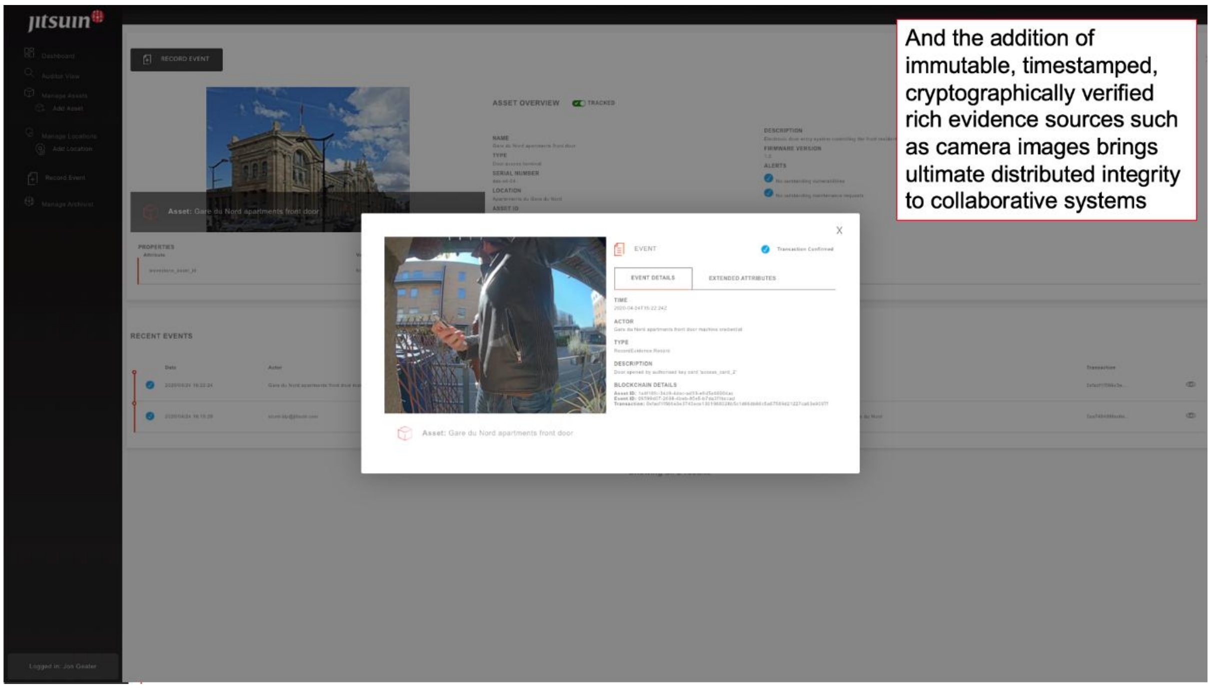
2019/09/04 05:33:57	51fcfaec-9a14-4b4a-9152-95c2b5ed4539@evc.m2m.synsation.io	RecordEvidence:Record	Device SJC-Large EV Charger-4 charging 26 units	0x9538f44101a4...
2019/08/28 10:04:31	51fcfaec-9a14-4b4a-9152-95c2b5ed4539@evc.m2m.synsation.io	RecordEvidence:Record	Device SJC-Large EV Charger-4 charging 61 units	0x85300282ebba...
2019/08/27 10:07:01	51fcfaec-9a14-4b4a-9152-95c2b5ed4539@evc.m2m.synsation.io	RecordEvidence:Record	Device SJC-Large EV Charger-4 charging 34 units	0x4339a23eaf0c...
2019/08/27 02:05:34	51fcfaec-9a14-4b4a-9152-95c2b5ed4539@evc.m2m.synsation.io	RecordEvidence:Record	Device SJC-Large EV Charger-4 charging 38 units	0x9e4659890fd7...
2019/08/26 23:49:09	ellaService@evcservicing.com	Firmware:Update	Responding to vulnerability 'CVE-2020-04-29 12:27:04.737196' with patch 'v7.1'	0xec37b4673dc4...
2019/08/25 21:40:58	51fcfaec-9a14-4b4a-9152-95c2b5ed4539@evc.m2m.synsation.io	RecordEvidence:Record	Device SJC-Large EV Charger-4 charging 53 units	0xecc8a047e2e3...
2019/08/25 01:44:45	51fcfaec-9a14-4b4a-9152-95c2b5ed4539@evc.m2m.synsation.io	RecordEvidence:Record	Device SJC-Large EV Charger-4 charging 26 units	0x69343d3028...
2019/08/24 19:38:06	51fcfaec-9a14-4b4a-9152-95c2b5ed4539@evc.m2m.synsation.io	RecordEvidence:Record	Device SJC-Large EV Charger-4 charging	0x0f9e59bcb72...
2019/08/24 18:34:29	**VulnBot@synsation-industries.com**	**Firmware:Vulnerability**	**Synsation Industries Large EV Chargers are vulnerable to CVE-2020-04-29 12:27:04.737196. Upgrade as soon as possible.**	**0x8379979dd13a...**
2019/08/24 15:46:13	Phil@evcservicing.com	Maintenance:Maintenance	Maintenance agent serviced device after 10146 units charged. Next service at 11000.	0x3ac74e84d77a...
2019/08/24 02:27:38	51fcfaec-9a14-4b4a-9152-95c2b5ed4539@evc.m2m.synsation.io	RecordEvidence:Record	Device SJC-Large EV Charger-4 charging 73 units	0xf2e427e02e10...
2019/08/23 09:13:03	51fcfaec-9a14-4b4a-9152-95c2b5ed4539@evc.m2m.synsation.io	RecordEvidence:Record	Device SJC-Large EV Charger-4 charging 41 units	0x51662 6cd2d38...
2019/08/22 13:17:32	**51fcfaec-9a14-4b4a-9152-95c2b5ed4539@evc.m2m.synsation.io**	**Maintenance:MaintenanceRequired**	**Service interval reached after 10032 units charged. Please service.**	**0x3f93e191258e...**

Adding telemetry and operations gives insight into correct servicing and use of devices, enabling digital business models or 'servitization' to operate with confidence

Vulnerability time-to-fix = 36h

Device used twice while in alarm condition

Maintenance time-to-fix = 48h



And the addition of immutable, timestamped, cryptographically verified rich evidence sources such as camera images brings ultimate distributed integrity to collaborative systems

EVENT

Transaction Confirmed

EVENT DETAILS | EXTENDED ATTRIBUTES

TIME
2020-04-24T15:22:242

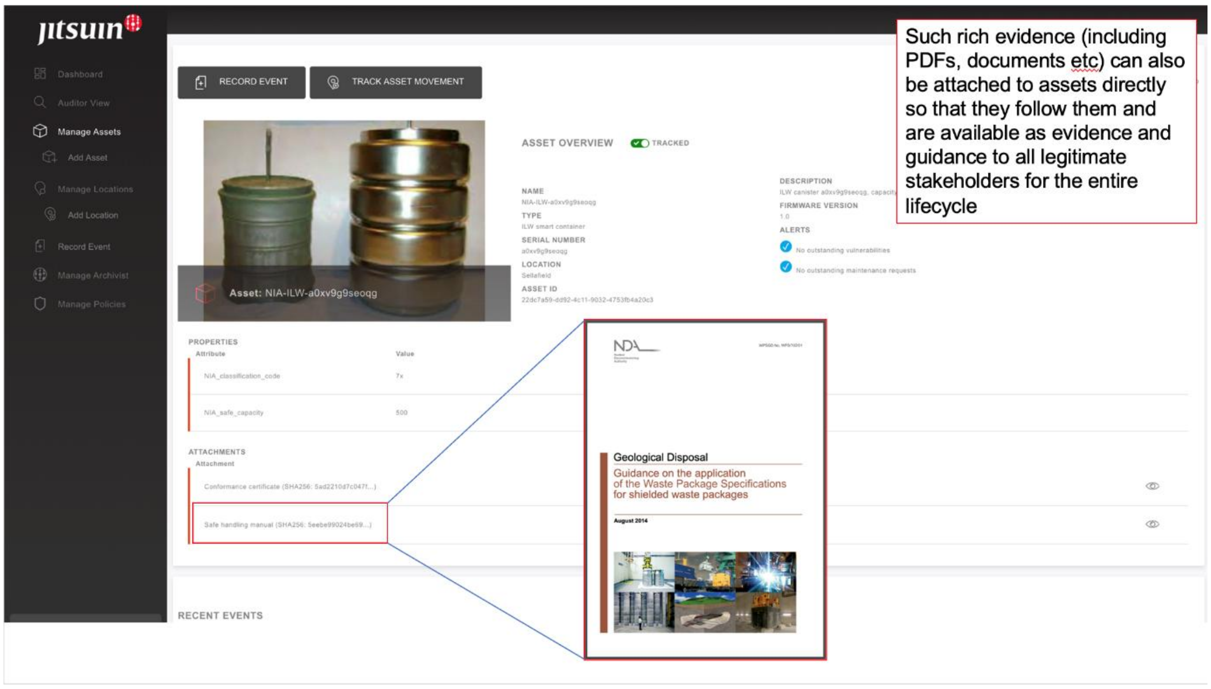
ACTOR
Gare du Nord apartments front door machine credential

TYPE
RecordEvidence Record

DESCRIPTION
Door opened by authorised key card 'access_card_2'

BLOCKCHAIN DETAILS
Asset ID: [illegible]
Event ID: [illegible]
Transaction: [illegible]

Asset: Gare du Nord apartments front door



Such rich evidence (including PDFs, documents etc) can also be attached to assets directly so that they follow them and are available as evidence and guidance to all legitimate stakeholders for the entire lifecycle

ASSET OVERVIEW

NAME
NIA-ILW-a0xv9g9seoqg

TYPE
ILW smart container

SERIAL NUMBER
a0xv9g9seoqg

LOCATION
Sellafield

ASSET ID
22dc7a59-dd92-4c11-9032-4753fb4a20c3

DESCRIPTION
ILW canister a0xv9g9seoqg, capacity

FIRMWARE VERSION
1.0

ALERTS
✓ No outstanding vulnerabilities
✓ No outstanding maintenance requests

PROPERTIES

Attribute	Value
NIA_classification_code	7x
NIA_safe_capacity	500

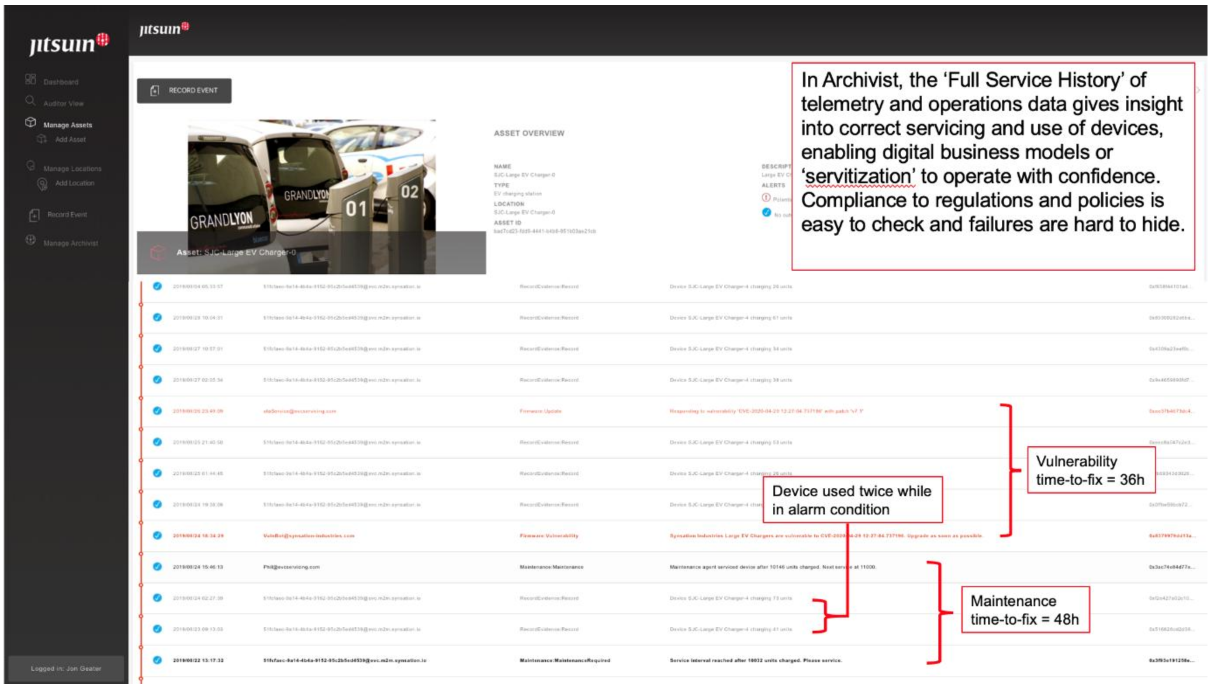
ATTACHMENTS

Attachment
Conformance certificate (SHA256: 5ad2210d7c0471...)
Safe handling manual (SHA256: 5eebe99024be69...)

WPSGD No. WPS/1001

Geological Disposal
Guidance on the application of the Waste Package Specifications for shielded waste packages

August 2014



NAME
SJC-Large EV Charger-0
TYPE
EV charging station
LOCATION
SJC-Large EV Charger-0
ASSET ID
bad7cd23-fdd5-4441-b4b8-851b03ae21cb

2019/08/24 15:46:13 Phil@evcservicing.com Maintenance:Maintenance Maintenance agent serviced device after 10146 units charged.

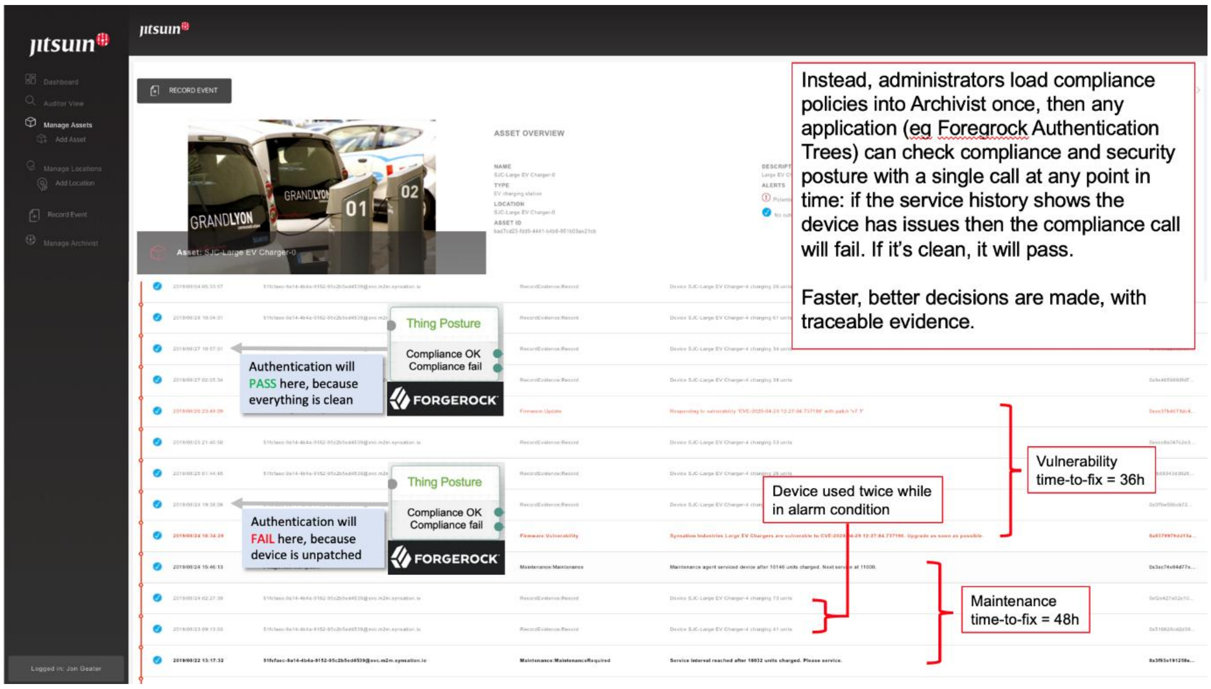
2019/08/22 13:17:32 51fcfaec-9a14-4b4a-9152-95c2b5ed4539@evc.m2m.synsation.io Maintenance:MaintenanceRequired Service interval reached after 10032 units charged. Please service.

In Archivist, the 'Full Service History' of telemetry and operations data gives insight into correct servicing and use of devices, enabling digital business models or 'servitization' to operate with confidence. Compliance to regulations and policies is easy to check and failures are hard to hide.

Vulnerability time-to-fix = 36h

Device used twice while in alarm condition

Maintenance time-to-fix = 48h



Instead, administrators load compliance policies into Archivist once, then any application (eg Foregrock Authentication Trees) can check compliance and security posture with a single call at any point in time: if the service history shows the device has issues then the compliance call will fail. If it's clean, it will pass.

Faster, better decisions are made, with traceable evidence.

Authentication will PASS here, because everything is clean

Authentication will FAIL here, because device is unpatched

Vulnerability time-to-fix = 36h

Device used twice while in alarm condition

Maintenance time-to-fix = 48h

Appendix H: A Case Study – Use Case ABS

Braking Control System Model

To verify the feasibility of the proposed stability analysis tool, an Anti-lock Braking System (ABS) has been selected for the case study.

Here the ABS model comes from MathWorks. Time delays are added into the system to mimic the network communication environment. The ABS controller is designed using Sliding Mode Control (SMC) technique to guarantee the robustness of the ABS.

Simulation Setup

A simulation model of the SMC-based ABS braking system has been built within MATLAB/Simulink environment as depicted in **Error! Reference source not found.**. Based on the definition of the r esearched attack types, a DoS – Deception attack simulation is implemented as depicted in **Error! Reference source not found.**.

Next the stability analysis tool is implemented to work in parallel with the ABS system (see Figure 32). The integrated simulation model is then obtained as shown in Figure 33**Error! Reference source not found.**. The tool will take the ABS control command and the feedback vehicle and wheel speeds as the inputs while provide four binary outputs, including:

- System stability (stable [1]/unstable [0])
- DoS attack flag (attack detected [1]/no attack [0])
- Deception attack flag (attack detected [1]/no attack [0])
- ML enable (enabled [1]/disabled [0]) – for the adaptability as discussed in Section **Error! R eference source not found.**

The model has been completely tested and then integrated with the blockchain to demonstrate its applicability (see the Blockchain report).

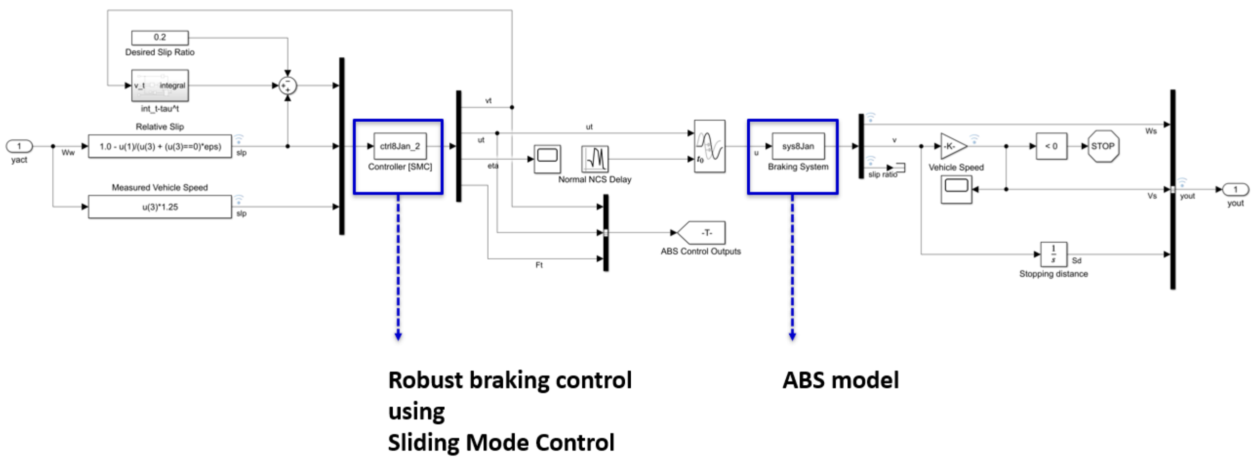


Figure 31 A case study - ABS control system using SMC

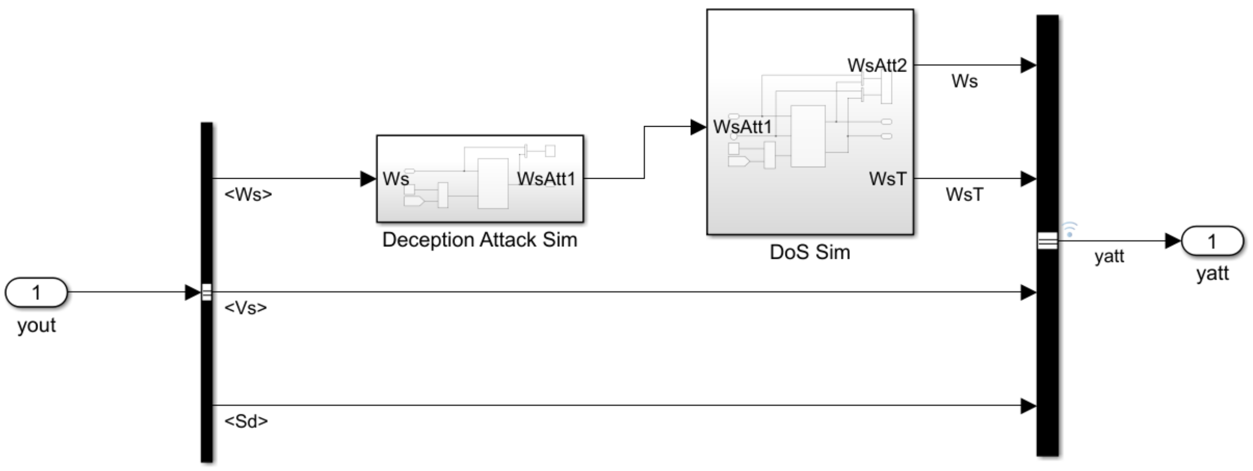


Figure 32 DoS and Deception attack simulator

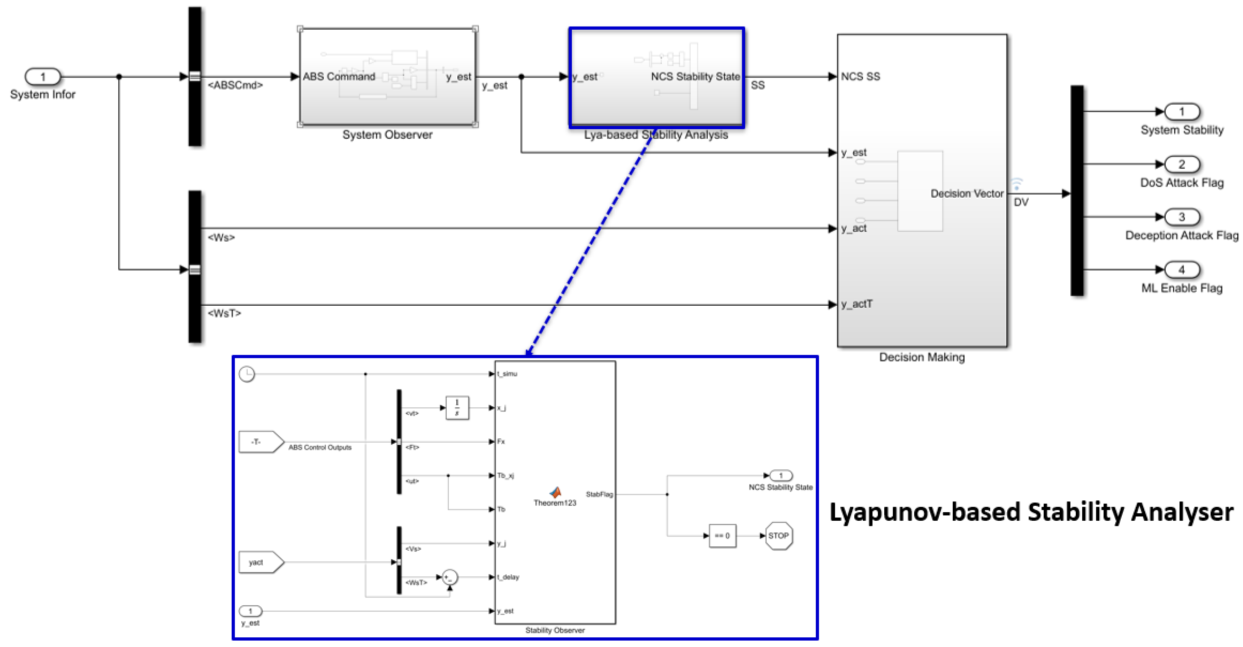


Figure 33 Stability analysis tool built within Simulink

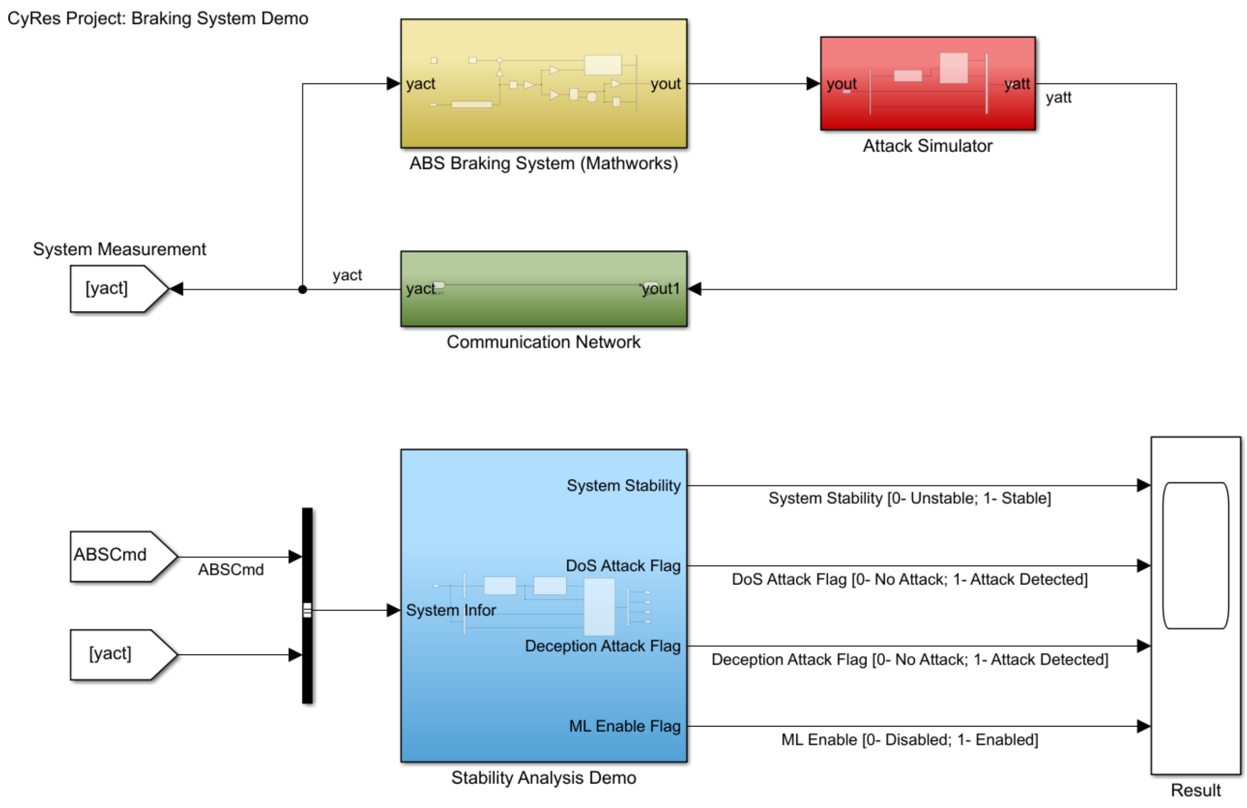

CyRes Project: Braking System Demo
yact
yout
ABS Braking System (Mathworks)
yout
yatt
yatt
Attack Simulator
System Measurement
[yact]
yact
yact
yout1
Communication Network
ABSCmd
ABSCmd
[yact]
System Stability
DoS Attack Flag
System Infor
Deception Attack Flag
ML Enable Flag
Stability Analysis Demo
System Stability [0- Unstable; 1- Stable]
DoS Attack Flag [0- No Attack; 1- Attack Detected]
Deception Attack Flag [0- No Attack; 1- Attack Detected]
ML Enable [0- Disabled; 1- Enabled]
Result

Figure 34 Integrated Simulink simulation program

To verify the feasibility of the proposed solution, the latter will be applied to the ABS use-case as in Figure 35. The top half is a representation of the real system, the variation in the performance of the communication network is an example of significant difference. This represents an instance of the real system; the bottom half is the *suitability and stability* analysis, as an instance (or result) of a simulation.

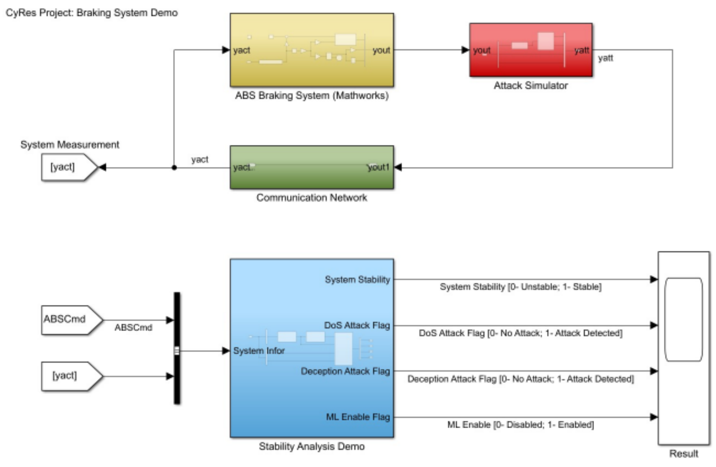

CyRes Project: Braking System Demo
yact
yout
ABS Braking System (Mathworks)
yout
yatt
yatt
Attack Simulator
System Measurement
[yact]
yact
yact
yout1
Communication Network
ABSCmd
ABSCmd
[yact]
System Stability
DoS Attack Flag
System Infor
Deception Attack Flag
ML Enable Flag
Stability Analysis Demo
System Stability [0- Unstable; 1- Stable]
DoS Attack Flag [0- No Attack; 1- Attack Detected]
Deception Attack Flag [0- No Attack; 1- Attack Detected]
ML Enable [0- Disabled; 1- Enabled]
Result

Figure 35: ABS model and simulator.

Phases

Recording events from Vehicle/Environment

The real system should report events to the blockchain whenever inputs or outputs are outside of those assumed or achieved by the simulation. For example, if the delay in the communications network (green box) is outside of the range assumed in the simulation this needs to be recorded as an event (Figure 36).

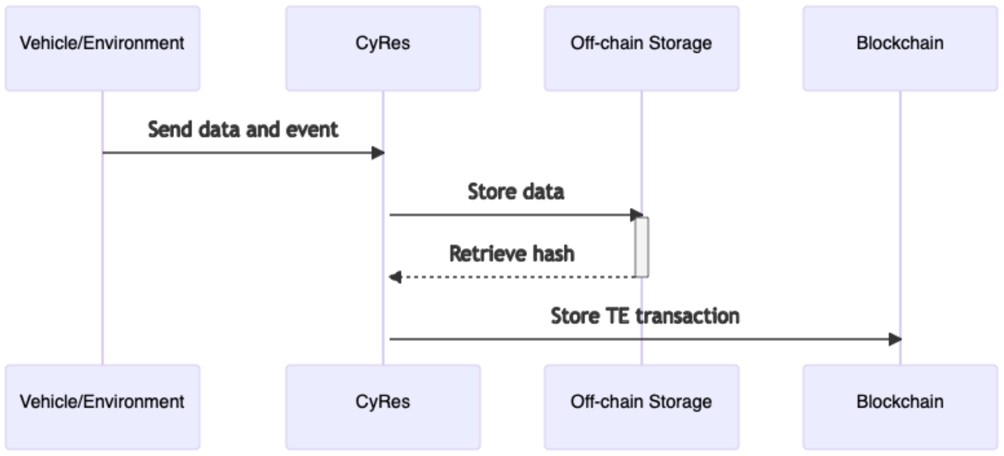


Figure 36: Creation of new events.

Preparing new simulations

A component in Cyres solution, namely the Monitor/Adjust block, processes new events and determine how up update the simulation to reflect those. In doing so, it generates multiple new sets of thresholds/constraints that are forwarded as input to the simulator(s) via the creation of simulation-request transactions (Figure 37).

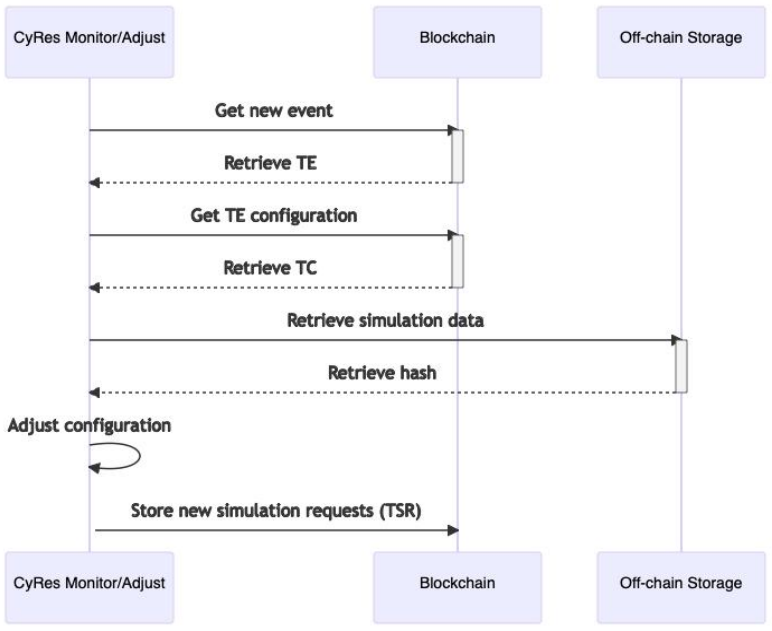


Figure 37: Preparation of multiple simulation requests.

Running simulations and storing outcomes

Each simulator block in the system can retrieve the corresponding simulation request transaction, i.e. the one with a compatible `subsystem` and `useCase` if any. Once completed, simulations declare if the system is stable or not and those outcomes are stored back into the distributed ledger (Figure 38).

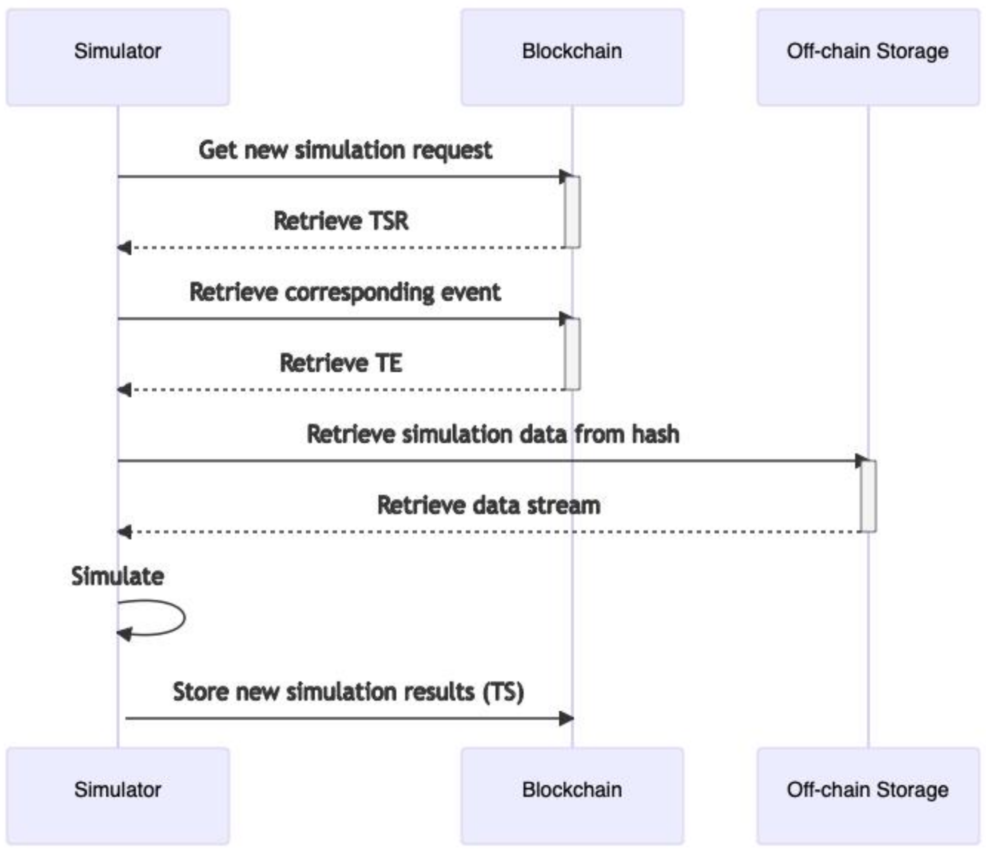


Figure 38: Concurrent simulation of the new conditions.

Evaluating simulation results

The re-running of the simulations with the updated constraints and assumptions provides a set of data for decisions to be made; with these results in the blockchain they are available for post assessment and validation or review. The decisions made as a result of assessing these results (either by human or machine) also needs to recorded in the distributed ledger sufficiently to identify the results being relied upon and decision reasoning. Using a smart contract, the Monitor/Adjust block collects all the simulations (TS) and, considering the outcome, it might decide to promote a new configuration, creating a TC transaction. The latter is stored into the distributed ledger and possibly forwarded to the vehicles, becoming the new baseline in identifying new events (Figure 39).

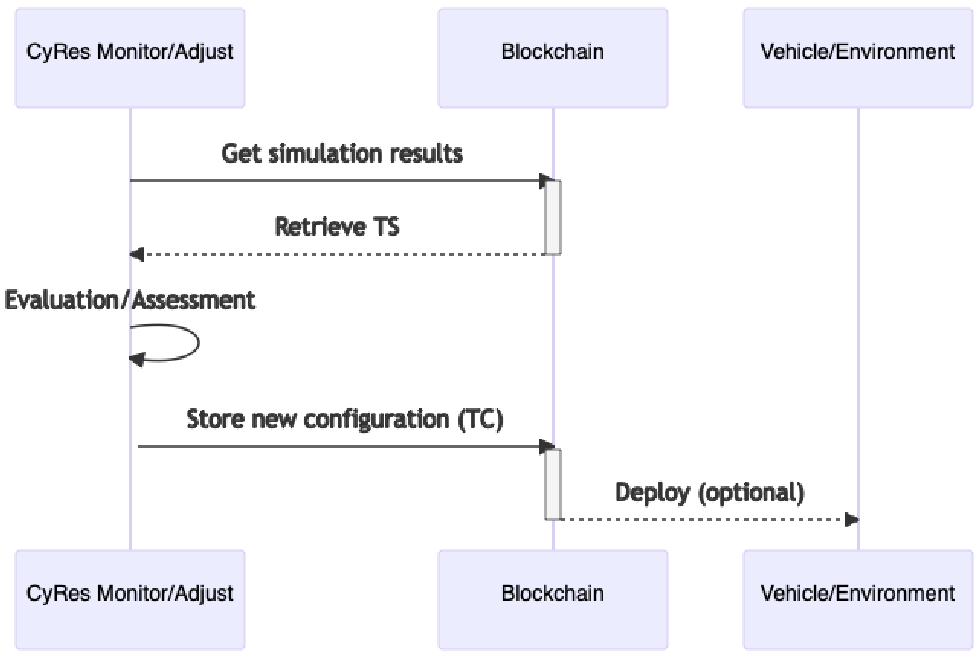


Figure 39: Evaluation of simulation results.

ZENZIC

SELF-DRIVING REVOLUTION

Insert call to action text here email@zenzic.io

zenzic.io