ESI

Powered by industry,
academia and TNO

MANAGING VARIABILITY AND EVOLUTION IN HIGH-TECH EQUIPMENT

FOSD 2024 Keynote

Prof. dr. Benny Akesson

Powered by industry, academia and TNO

ESI

KEY MESSAGES

This presentation has four key messages:

1. System complexity trends for high-tech equipment

2. TNO-ESI and its role in the high-tech industry

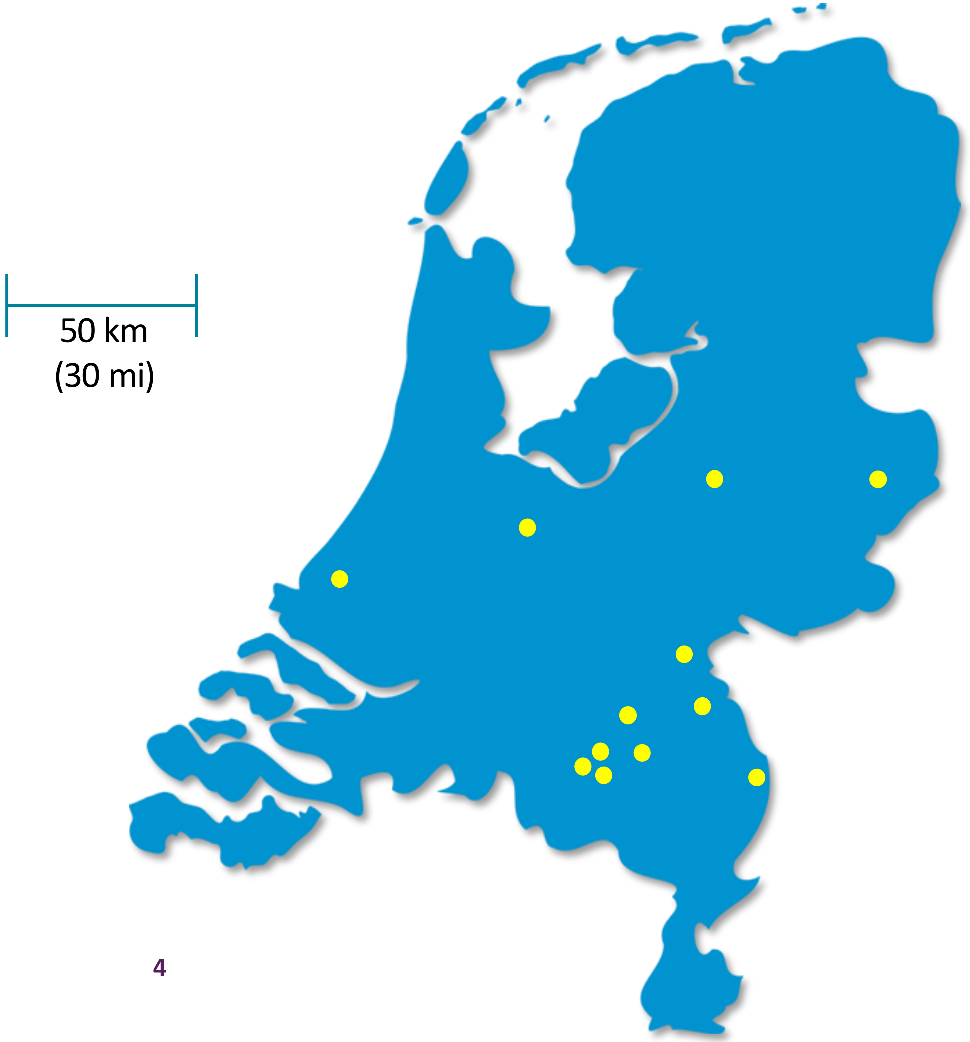
3. Finished industry case:
   Specification, verification, and adaptation of software interfaces

4. Ongoing industry case:
   Variability and evolution in software platforms

ESI

Powered by industry,
academia and TNO

EXAMPLES OF DUTCH HIGH-TECH SYSTEMS



SYSTEM COMPLEXITY IS INCREASING!

- Five technological and market trends drive increasing complexity in high-tech systems:

1. **Additional functionality**
   - Number of interfaces and lines of code are **rapidly increasing**
2. **Mass customization**
   - Increased customization of systems at design time to the point where **each system is unique**
3. **Long life times**
   - Systems operate for decades and need to **continuously evolve** after deployment
4. **Increasing autonomy**
   - Systems acting autonomously with **little or no human interaction**
5. **Systems of systems**
   - Interconnected systems of which **nobody is in complete control**

MANAGING COMPLEXITY

- Managing complexity in high-tech systems is critical to successful development and deployment
  - Impacts all phases of development: **design**, **implementation**, **verification**, and **evolution**

- Increasing complexity cannot be dealt with by current engineering methodologies
  - Increasing **development and maintenance costs**
  - Increasingly hard to **guarantee functional correctness** and **balance system qualities**
  - Severe shortage of **skilled people**

- **New design methodologies** are required to manage the increasing complexity and enable future generations of systems to be developed efficiently!

- ESI is an organization that **orchestrates** the **innovation chain** for design methodologies in the Dutch high-tech eco-system and conducts **applied research to improve industrial practice**

ESI

Powered by industry, academia and TNO

TNO-ESI AT A GLANCE

SYNOPSIS

- Foundation ESI started in 2002
- ESI acquired by TNO per January 2013
- ~60 staff members many with extensive industrial experience
- 8 part-time professors

FOCUS

Managing complexity of high-tech systems

through

- system architecting
- system reasoning and
- model-driven engineering

delivering

- methodologies validated in cutting-edge industrial practice

PARTNER BOARD

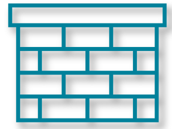
ASML, Canon – Canon Production Printing, itec – equipment + automation tech, Philips, Radboud University Nijmegen, Thales – Building a future we can all trust, TNO, TU Delft, TU/e Eindhoven University of Technology, University of Twente, University of Amsterdam, Vanderlande

Capgemini engineering, Strypes

Powered by industry, academia and TNO

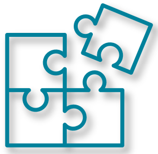
ABC OF COMPLEXITY MANAGEMENT

A. **Abstraction**: Identify high-level concepts that hide low-level details
   - Software is programmed in high-level programming languages and not using machine code



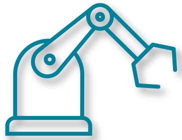
B. **Boundedness**: Impose acceptable restrictions on the considered problem space
   - Constraints on environment in which system has to function correctly

C. **Composition**: Divide one problem into multiple independent smaller problems
   - System is decomposed into logical functions that are developed individually



Automation is key to coping with complexity

- Not a fundamental technique to reduce complexity, rather about **productivity**



MANAGING COMPLEXITY WITH MODELS

- **Model-based methodologies** are promising for managing complexity

- There is a clear relation to the ABC Complexity Management Techniques
  - Models are **abstractions** of the system
  - System **boundedness** can be expressed through parameter ranges and constraints in models
  - Systems are (de-)composed into models covering different parts or aspects.
    Model-based development methodologies are the **composition** of these

- The formal nature of models provide a strong link to **automation**
  - Models are used as a **source** for automated **analysis** and **synthesis**

2

SPECIFICATION, VERIFICATION, AND ADAPTATION OF SOFTWARE INTERFACES

2019-2021

MOTIVATION

- Thales systems have life time > 30 years and require upgrades
  - Both technology and customer requirements **significantly change** during life time
  - Compute nodes need to be added or replaced to **counter obsolescence** or **improve performance**
  - **New software with new capabilities** becomes available

- System upgrades can take 1-2 years and happen every 10-15 years
  - Many small updates collected into **big infrequent upgrades**
  - System **evolves slowly** and in big steps, increasing risk

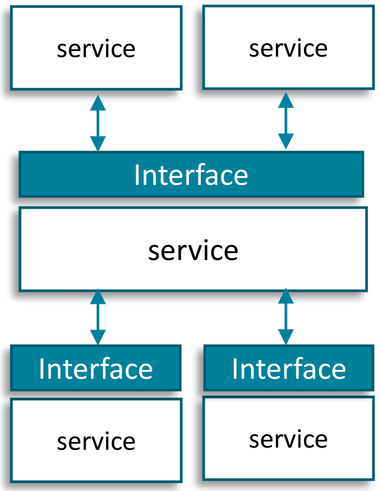
- Systems need to continuously evolve to **reduce risk** and **increase added value**

MODULARITY IN SOFTWARE ARCHITECTURES

ESI
Powered by industry, academia and TNO

- Increasing software complexity is tackled by **modularization**
    - Software is **decomposed** into **services** corresponding to particular functionality **(composition)**
    - **Asynchronous communication** often used to achieve **loose coupling** between services
    - **Abstraction** of service implementation and technology is provided by an **interface**

- Modularity addresses the stated complexity drivers
    - Improved **scalability**, **customization**, and **evolvability**



PROBLEM STATEMENT

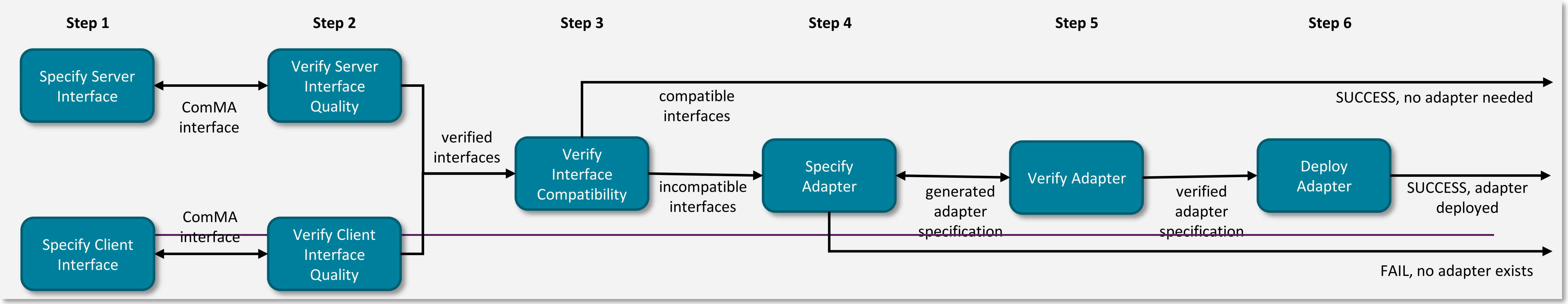
Powered by industry, academia and TNO

1. Many asynchronously communicating services lead to an explosion of possible behaviors
    - Interactions between services become **very hard to verify**
    - **Early design errors** are detected much later in the system life-cycle, **increasing cost**
2. Updating interfaces becomes prohibitive
    - **Manually changing an interface is quick**, but **updating many incompatible services takes time**
    - Evolving interfaces is hence **expensive** and **time consuming**, resulting in **technical debt**

METHODOLOGY OVERVIEW

ESI
Powered by industry, academia and TNO

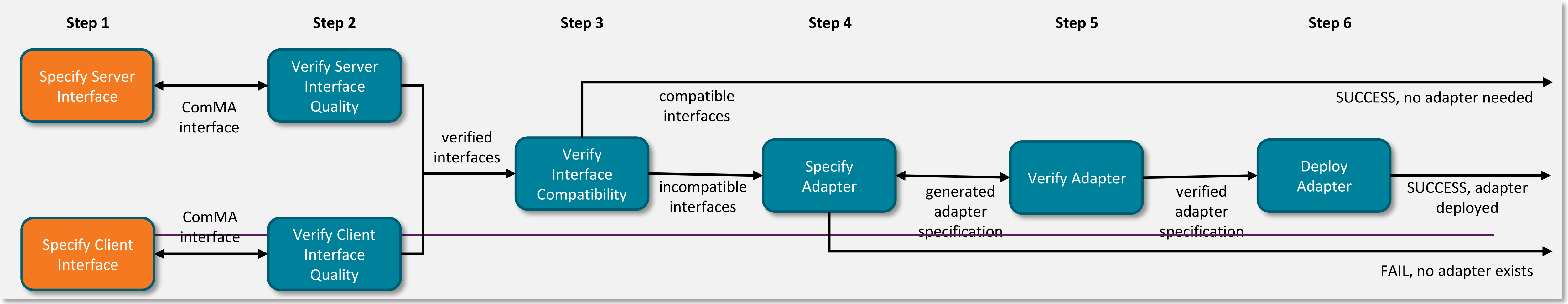
- Six-step Methodology
  1. **Specify structure and behavior of interfaces** using **ComMA**
  2. **Verify quality of interfaces** using **quality dashboard** and resolve issues through **design guidelines**
  3. **Verify structural and behavioral compatibility** between server and client interfaces
  4. **Specify adapter** between client and server using **Mapping DSL**
  5. **Verify compatibility** of server and client using **generated adapter**
  6. **Deploy Adapter** in the system



METHODOLOGY OVERVIEW

ESI
Powered by industry, academia and TNO

- Six-step Methodology
  1. **Specify structure and behavior of interfaces** using **ComMA**
  2. **Verify quality of interfaces** using **quality dashboard** and resolve issues through **design guidelines**
  3. **Verify structural and behavioral compatibility** between server and client interfaces
  4. **Specify adapter** between client and server using **Mapping DSL**
  5. **Verify compatibility** of server and client using **generated adapter**
  6. **Deploy Adapter** in the system



INTERFACE SPECIFICATION

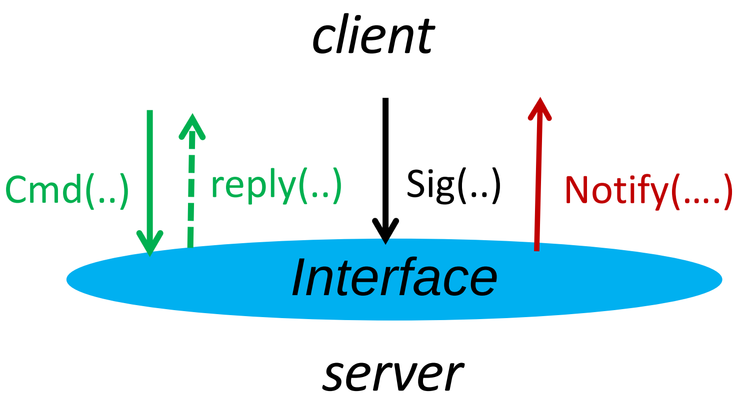
- ComMA was selected as specification language for six good reasons
    1. **We did not want to reinvent the wheel** by making a new interface specification language
    2. specifies **both structure and behavior**, required to validate both aspects of compatibility
    3. models **both synchronous and asynchronous** communication
    4. **automatic inference** and **migration** of interface specifications simplifies industrial adoption
    5. the tooling is based on **Eclipse**, which is one of the most commonly used modeling tools in the embedded domain
    6. developed and **successfully applied in industry**

Web: https://projects.eclipse.org/projects/technology.comma

ABOUT ECLIPSE COMMASUITE

Domain Specific Language (DSL) to express client-server interface:

- **Signature**
  - Commands: synchronous function calls - from client to server
  - Signals: asynchronous function calls - from client to server
  - Notifications: asynchronous notifications - from server to client(s)
- **Behavior** by **protocol state machine**
  - Contract between client(s) and server; allowed sequences of events
  - Non-determinism allowed
- Supports both **data constraints** and **timing constraints**



BIGGER PICTURE WITH COMMA

ESI
Powered by industry, academia and TNO

Model Inference

Structure

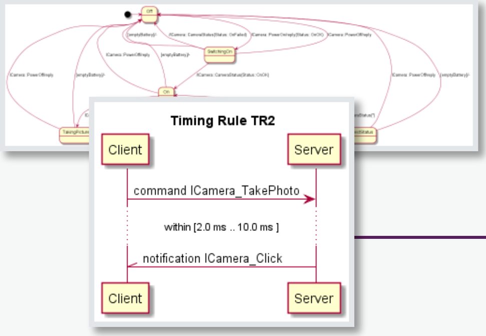
```
commands
    Status PowerOn
    void   PowerOff
    bool Zoom(in real factor,
              out real result,
              out real scale
    )
    Picture TakePhoto(Time t)
    int  GetPictureNumber
signals
    ProvideStatus
notifications
    CameraStatus(Status s)
    Click
}
```

Behavior

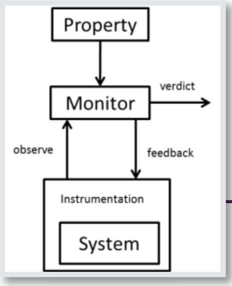
```
initial state Off {
    transition trigger: ICamera::PowerOn
        do: reply(Status::OnOK)          next state: SwitchingOn
        OR
        do: reply(Status::OnFailed)      next state: Off
}
state SwitchingOn {
    transition do: ICamera::CameraStatus(Status::OnOK)
                                                next state: On

    transition do: ICamera::CameraStatus(Status::OnFailed)
                                                next state: Off
}
```

Generated

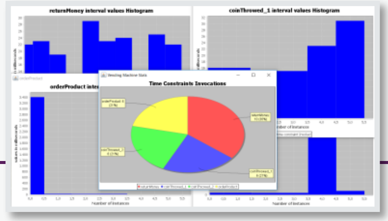
Visualization



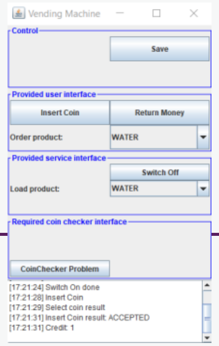
Monitoring



Statistics



Simulation

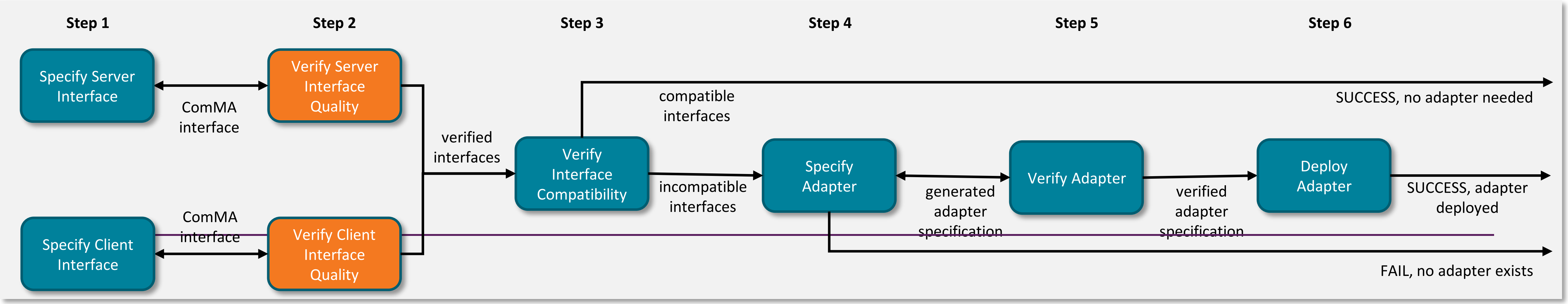


Documentation

METHODOLOGY OVERVIEW

- Six-step Methodology

  1. **Specify structure and behavior of interfaces** using **ComMA**
  2. **Verify quality of interfaces** using **quality dashboard** and resolve issues through **design guidelines**
  3. **Verify structural and behavioral compatibility** between server and client interfaces
  4. **Specify adapter** between client and server using **Mapping DSL**
  5. **Verify compatibility** of server and client using **generated adapter**
  6. **Deploy Adapter** in the system



REACHABILITY GRAPH

- Verification of ComMA specification possible by translation into **Colored Petri Net (CPN)** model

- State of a CPN model
  - **Full state** is the **marking** + **values of all variables**
  - **Initial full state** is **initial marking** + **initial values of all variables**

- A **reachability graph** is generated from the CPN model
  - Considers **all paths** from the initial full state for a given set of input data

MODEL QUALITY CHECK

- Reachability graphs model quality checking using reachability analysis
  - Allows **state and transition coverage** of interface to be determined
  - Enables **unreachable states**, **deadlocks**, **livelocks**, and **sink states** in server state machine to be detected
  - Also **race conditions**, **property violations**, and **confusion** in interactions with a **mirrored client**

- User gets feedback on model quality from a dashboard
  - **Lists** and **visualizes** quality issues and provides **guidelines** for how to resolve them

IMPRESSION OF MODEL QUALITY DASHBOARD

Powered by industry, academia and TNO

Statistics

Reachability Graph Coverage:	Full [4 states]
Transition Coverage:	100.0%
State Coverage:	100.0%
Num Of States:	4

Summary

Interface

Server State Machine

- Deadlocks (0)
- Livelocks (0)
- Unreachable States (0)
- Sink States (0)

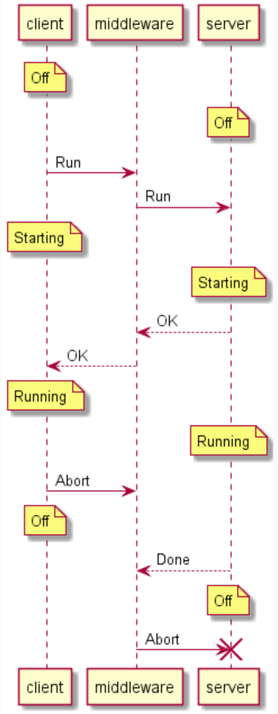
Client Communication

- Choice Properties (2)
- Leg Properties (7)
- Service Race Conditions (1)
- Client Race Conditions (0)
- Simple Confusions (0)

Verification Results of ComMA Interface

ServerRaceConditions

Race Condition in State: Running



Resolution: When server emits notifications : Done - Signal Abort is missing in state Off

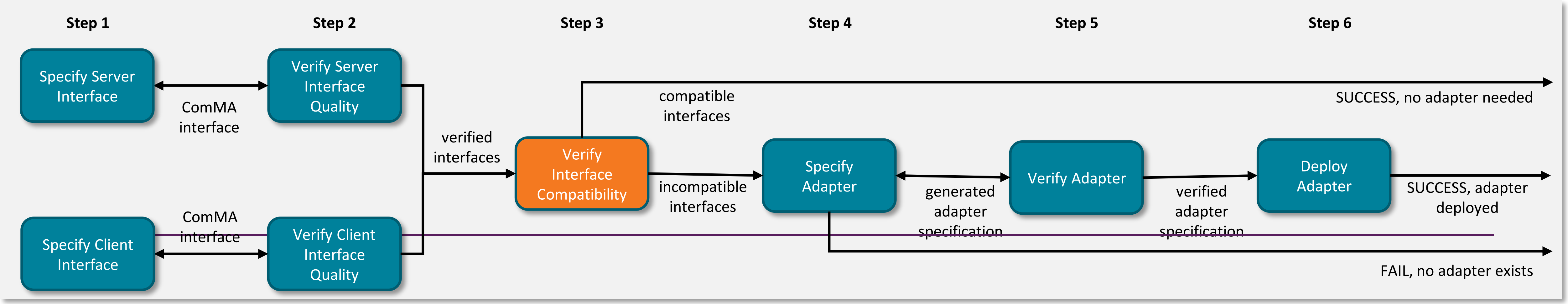
```
import "Example.signature"

interface Example version "1.0"

machine StateMachine {

        initial state Off {
                transition trigger: Run next state: Starting
                transition do: Done next state: Off
        }

        state Starting {
                transition do: OK next state: Running
                transition do: Done next state: Off
        }

        state Running {
                transition do: Done next state: Off
                transition trigger: Run next state: Rejecting
                transition trigger: Abort next state: Off
        }

        state Rejecting {
                transition do: NOT_ALLOWED next state: Running
                transition do: Done next state: Off
        }
}
.ÓMÝ
```

METHODOLOGY OVERVIEW

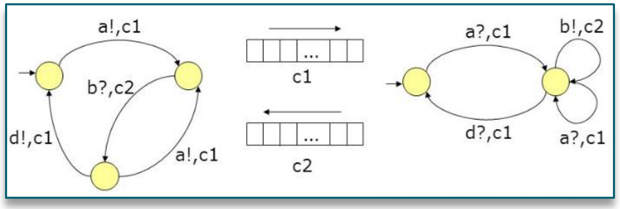
ESI
Powered by industry, academia and TNO

- Six-step Methodology

  1. **Specify structure and behavior of interfaces** using **ComMA**
  2. **Verify quality of interfaces** using **quality dashboard** and resolve issues through **design guidelines**
  3. **Verify structural and behavioral compatibility** between server and client interfaces
  4. **Specify adapter** between client and server using **Mapping DSL**
  5. **Verify compatibility** of server and client using **generated adapter**
  6. **Deploy Adapter** in the system

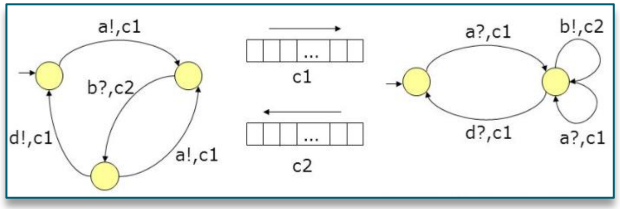


Client ComMA model



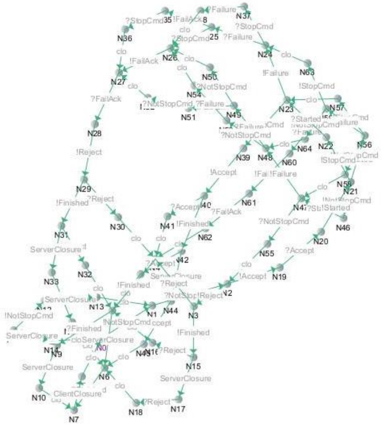
Petri net

Server ComMA model



Petri net

Connect nets and export to PNML file



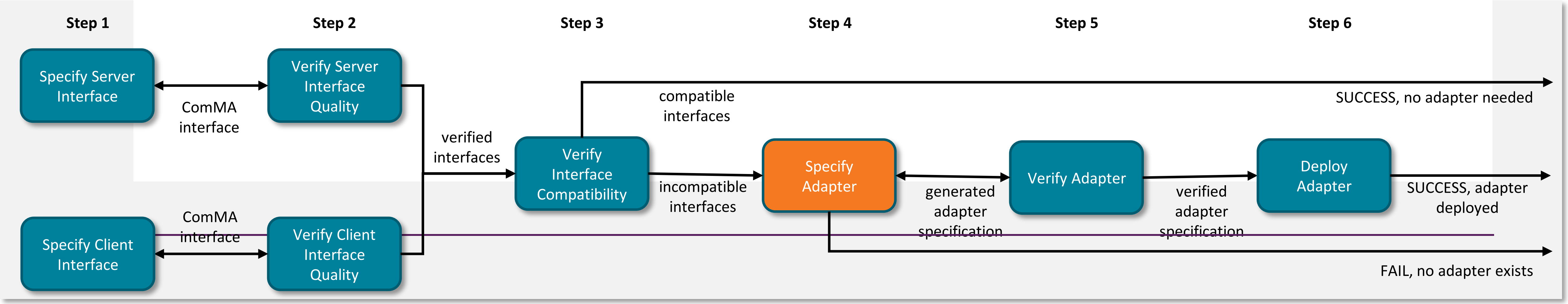
Use open-source Petri net Analysis Tools (PnAT) to compute reachability graph and check for termination (does not consider data)

```
**************************************************
  Property Check for Weak Termination
**************************************************

> Checking for Boundedness..

[ Summary ]

<+> Weak Termination: OK
<+> The net is safe!
**************************************************
```

METHODOLOGY OVERVIEW

- Six-step Methodology

1. **Specify structure and behavior of interfaces** using **ComMA**
2. **Verify quality of interfaces** using **quality dashboard** and resolve issues through **design guidelines**
3. **Verify structural and behavioral compatibility** between server and client interfaces
4. **Specify adapter** between client and server using **Mapping DSL**
5. **Verify compatibility** of server and client using **generated adapter**
6. **Deploy Adapter** in the system



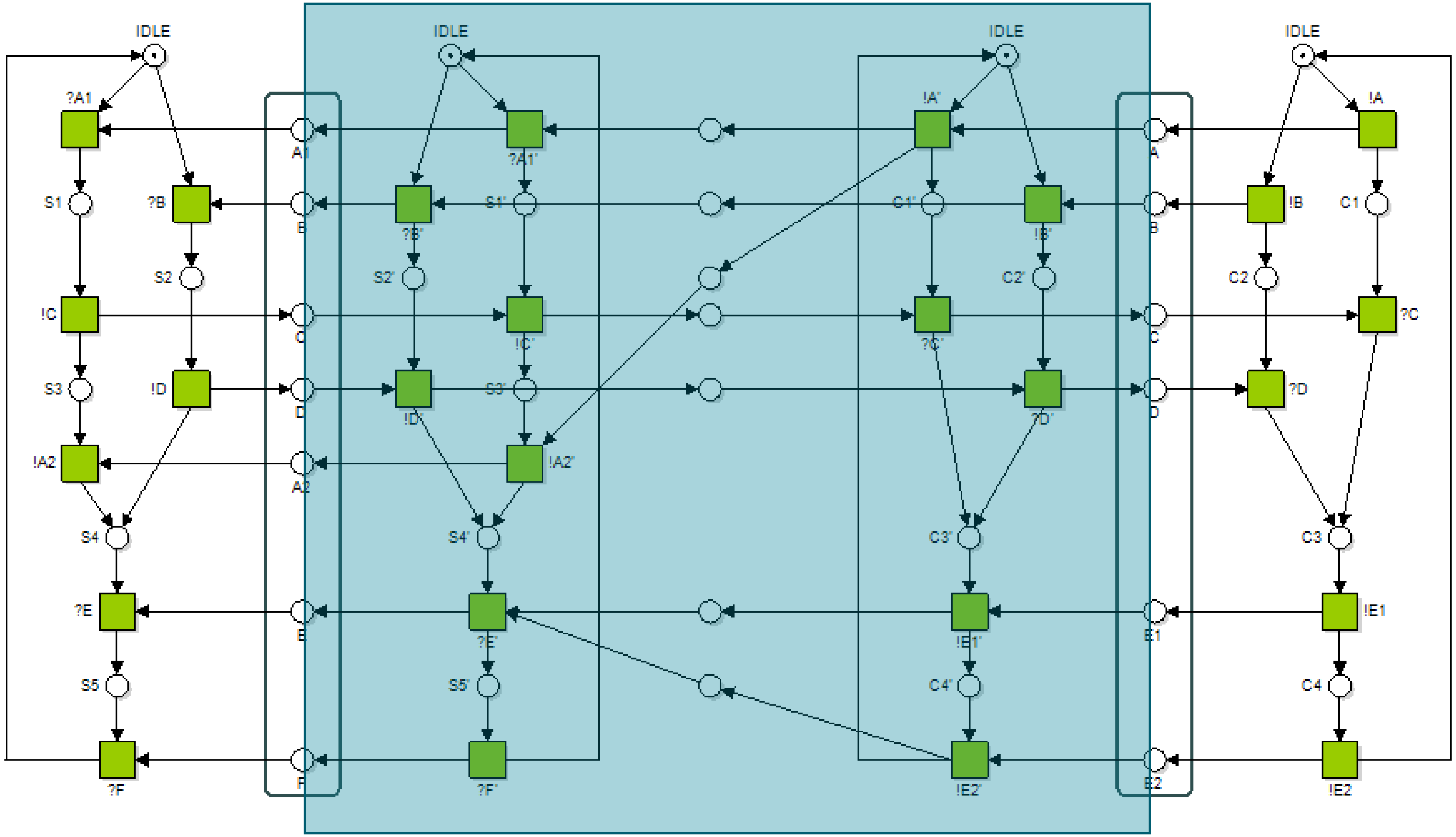
GENERATE ADAPTER

- Overview of adapter generation approach
  - **ComMA Mapping DSL** describes relation between messages for server and client
  - Adapter **encodes mapping rules** and is responsible for **transformations**
  - DSL generates **adapter Petri net specification**

Mapping Rules

cs.PTOn -> sr.PTOn;
cs.PTGetState -> sr.PTGetState;
cs.PTOff -> sr.PTOff, cr.PTState;
ss.PTState -> cr.PTState;

SERVER AND CLIENT CONNECTED THROUGH AN ADAPTER



Mapping Rules

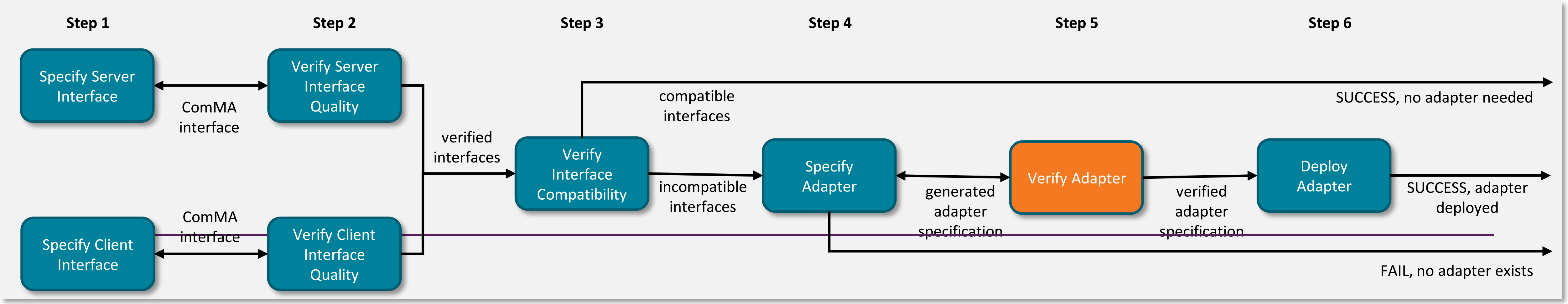
cs.A -> sr.A1, sr.A2
cs.B -> sr.B
ss.C -> cr.C
ss.D -> cr.D
cs.E1, cs.E2 -> sr.E
-> sr.F

METHODOLOGY OVERVIEW

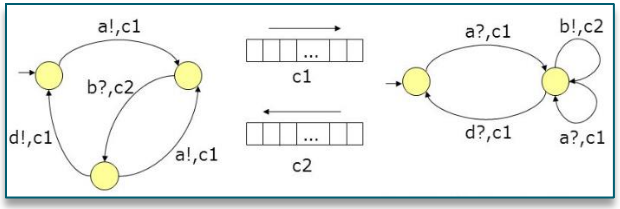
Powered by industry, academia and TNO

- Six-step Methodology

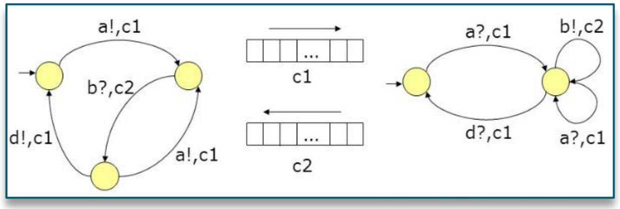
1. **Specify structure and behavior of interfaces** using **ComMA**
2. **Verify quality of interfaces** using **quality dashboard** and resolve issues through **design guidelines**
3. **Verify structural and behavioral compatibility** between server and client interfaces
4. **Specify adapter** between client and server using **Mapping DSL**
5. **Verify compatibility** of server and client using **generated adapter**
6. **Deploy Adapter** in the system



Client ComMA model



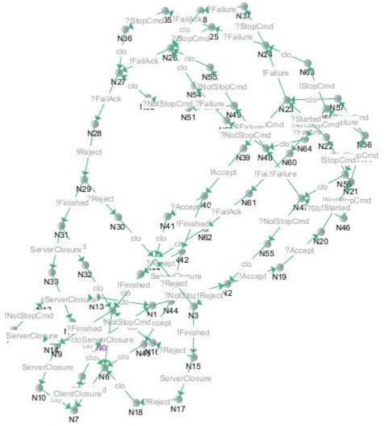
Server ComMA model



ComMA Mapping DSL

Petri net | **Petri net** | **Petri net**

Connect nets and export to PNML file



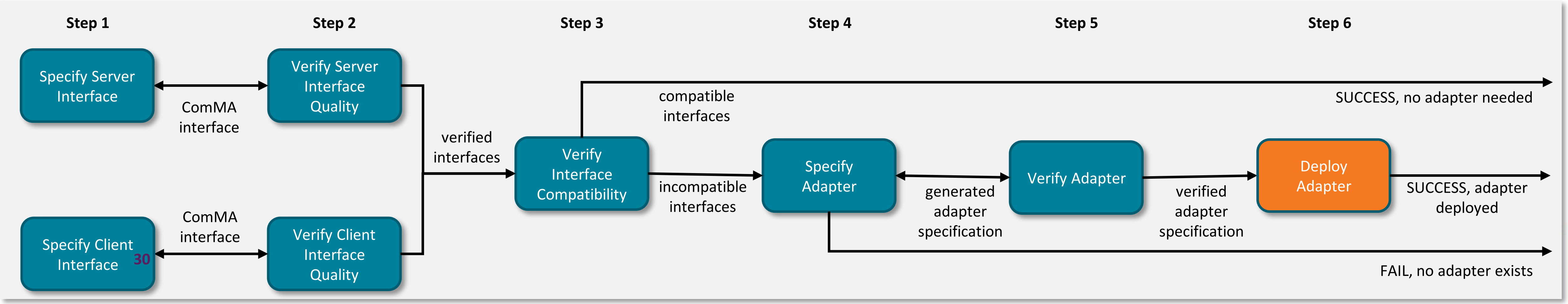
Use open-source Petri net Analysis Tools (PnAT) to compute reachability graph and check for termination (does not consider data)

```
****************************************************
  Property Check for Weak Termination
****************************************************


> Checking for Boundedness..

[ Summary ]

<+> Weak Termination: OK
<+> The net is safe!
****************************************************
```

METHODOLOGY OVERVIEW

- Six-step Methodology

1. **Specify structure and behavior of interfaces** using **ComMA**
2. **Verify quality of interfaces** using **quality dashboard** and resolve issues through **design guidelines**
3. **Verify structural and behavioral compatibility** between server and client interfaces
4. **Specify adapter** between client and server using **Mapping DSL**
5. **Verify compatibility** of server and client using **generated adapter**
6. **Deploy Adapter** in the system



BIGGER BIGGER PICTURE WITH COMMA

Model Inference →

Structure

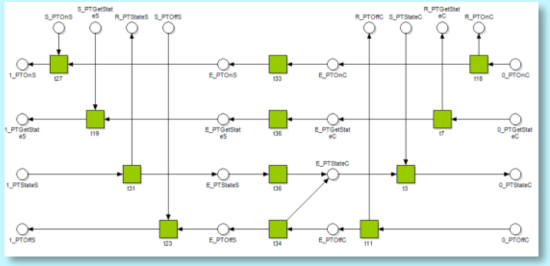
```
commands
    Status PowerOn
    void   PowerOff
    bool Zoom(in real factor,
              out real result,
              out real scale
    )
    Picture TakePhoto(Time t)
    int  GetPictureNumber
signals
    ProvideStatus
notifications
    CameraStatus(Status s)
    Click
}
```

Behavior

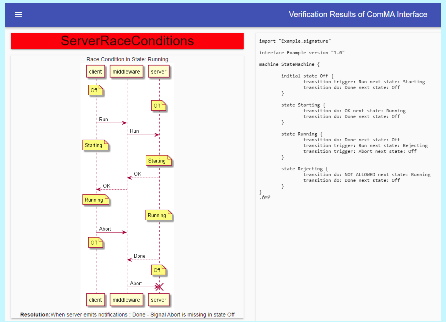
```
initial state Off {
    transition trigger: ICamera::PowerOn
        do: reply(Status::OnOK)            next state: SwitchingOn
        OR
        do: reply(Status::OnFailed)        next state: Off
}
state SwitchingOn {
    transition do: ICamera::CameraStatus(Status::OnOK)
                                                   next state: On

    transition do: ICamera::CameraStatus(Status::OnFailed)
                                                   next state: Off
}
```

Generated →

Adapter Generation



Model Quality Checks

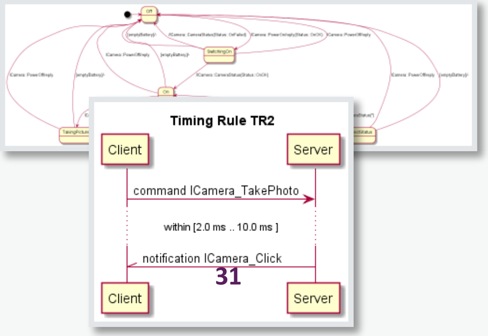


Code Stubs

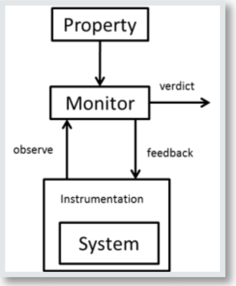
```
switch(state) {
  case ON {

  }
  case OFF{

  }
}
```

Generated ↓

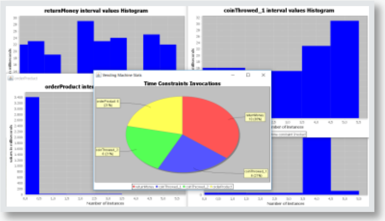
Visualization



Monitoring



Statistics



Simulation

Documentation

OPEN INNOVATION AT WORK!

Powered by industry, academia and TNO

- This project is a text book example of **open innovation** at work
    - Initial prototype was based on theory and tools from previous European academic research
    - Leveraged Eclipse CommaSuite (ESI&Philips) and results **were contributed back to open source**
    - **Methodology** and **proof-of-concept tool** delivered to Thales to **evaluate for adoption**
    - **Three UvA students** contributed to the research, publishing **theses** and **academic papers**

- Developed knowledge transferred in a **two-day course**
    - Two versions of course, one based on **Petri nets** and one on **CommaSuite**
    - Course has been given to >300 participants at Thales and UvA

3

VARIABILITY AND EVOLUTION IN SOFTWARE PLATFORMS

2024-?

INTRODUCTION

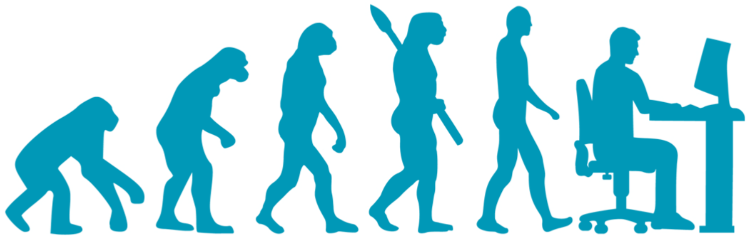
- There is a trend towards **increased customization** in many industries, including high-tech equipment
  - Every customer wants slightly different features
  - Puts pressure on **R&D effort**, **delivery times,** and **cost** of products

- Industry has been addressing this challenge by moving towards a **platform-based approach**
  - New **customized products** can be derived by configuring a general set of **building blocks**
  - Often requires a transition from project organizations to **product organizations**
  - From **engineering-to-order** to **configure-to-order**

LONG LIFE TIMES AND CONTINUOUS EVOLUTION

- High-tech equipment has long life times of **several decades** in the field
  - Need to cope with end of life of components, spare parts, and maintenance
  - The system **outlives** many of its constituent technologies, in particular digital technologies

- There is a need for the system to **continuously evolve** during its life time
  - System needs upgrades to **new (digital) technology and software** to remain operational
  - Customer wants **improved functionality** that increase the added value



PROBLEM STATEMENT

ESI
Powered by industry, academia and TNO

- Evolving system variants requires substantial **re-development effort** and is **costly**
  - Essentially a new iteration through the complete development cycle
  - Product configurations must be **updated** to reflect new features
  - System is often **manually ported** to new software technologies
  - System must be **re-verified**, including its **performance**, which is an emerging property from interacting hardware and software components

- This project addresses the challenge of **reducing the time and cost** associated with **system evolution** at the level of the **software platform** (infrastructure)

VISION AND GOAL

Vision

A **product-based approach** with **building blocks** to quickly **configure custom solutions** for each customer that **always satisfy their (performance) requirements** throughout their **entire lifecycle**

Long-term goal

A **model-based methodology** and supporting tool that enables custom solutions to be **specified in a technology-agnostic manner**, and where a software deployment that **satisfies (performance) requirements** is **automatically generated** and regenerated, as software technologies evolve

RESEARCH QUESTIONS AND APPROACH FOR 2024

Powered by industry, academia and TNO

1. To what extent can the generic configuration model be **decoupled** from the deployment technology?
    - Investigating a **domain-specific language** that describes system modules at the level of the capabilities the provide and require, and the hardware and software components they need to realize functionality
    - Specification is **independent** from the software technology used, e.g. for containerization or orchestration, such as Kubernetes or Docker
    - Automatic generation of deployments for different technologies addresses evolution of the **level of the platform**, reducing effort for all variants

2. How can we efficiently **identify a software deployment** that satisfy performance requirements for a particular product configuration and deployment technology?
    - Investigating a **learning-based approach** to optimize deployments for a **particular variant**
    - Performance verification successful if a (set of) deployment can be found that satisfies requirements for relevant loads

4

CONCLUSIONS

CONCLUSIONS

ESI
Powered by industry, academia and TNO

- The system complexity of high-tech equipment is increasing
  - Market demands for **more functionality**, **customization**, and **evolvability**
  - ESI and its partners addresses this complexity challenge using **model-based systems engineering methodologies** in an **open innovation ecosystem**

- **Two industrial cases** related to variability and evolution of software in high-tech equipment were presented
  1. A model-based methodology for specification, verification, and adaptation of software interfaces based on domain-specific languages and Petri nets was presented and its impact discussed
  2. Problem description and research directions for a challenge to reduce the cost and effort related to variability and evolution at the level of the software platform

- We are happy to discuss this research with you and hear your **feedback** and **input**
  - … and let us know if you are interested in making the **next step** in your career ☺

ACKNOWLEDGEMENTS

THE RESEARCH IS CARRIED OUT AS PART OF THE DYNAMICS AND TECHFLEX PROJECTS UNDER THE RESPONSIBILITY OF TNO-ESI WITH THALES NEDERLAND B.V. AS THE CARRYING INDUSTRIAL PARTNER. THE DYNAMICS AND TECHFLEX RESEARCH IS SUPPORTED BY THE NETHERLANDS ORGANISATION FOR APPLIED SCIENTIFIC RESEARCH TNO.

Powered by industry, academia and TNO