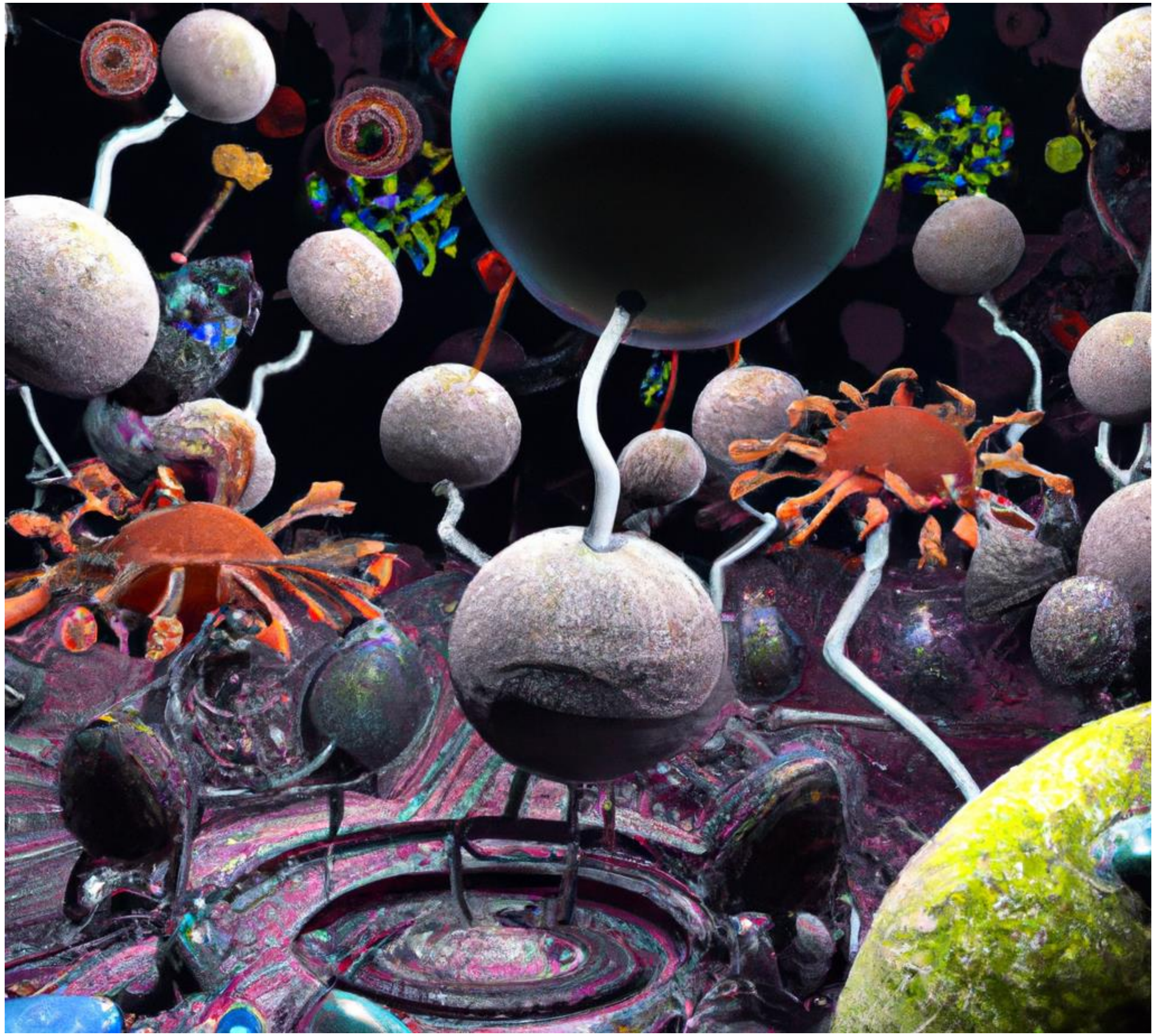
Organic Systems of Systems (OSoS): State of the Art Modeling and Beyond

Markus Schacher, KnowBody

Hohlstrasse 534, 8048 Zürich, Switzerland, www.knowgravity.com

Organic Systems of Systems (OSoS)



An Organic System of Systems (OSoS) is a **long lasting, large and fuzzy ecosystem** consisting of many dynamic, **adaptive and potentially autonomous sub-systems** (including various life forms), each individually composed of its own sub-systems and all together collaboratively **pursuing a common purpose** within a highly dynamic environment.

Examples of OSoS

(Smart) Buildings

Transportation Systems

Digital Enterprises

Power Grids

Complex Plants

Distributed Defence

(Smart) Cities

Properties of an OSoS

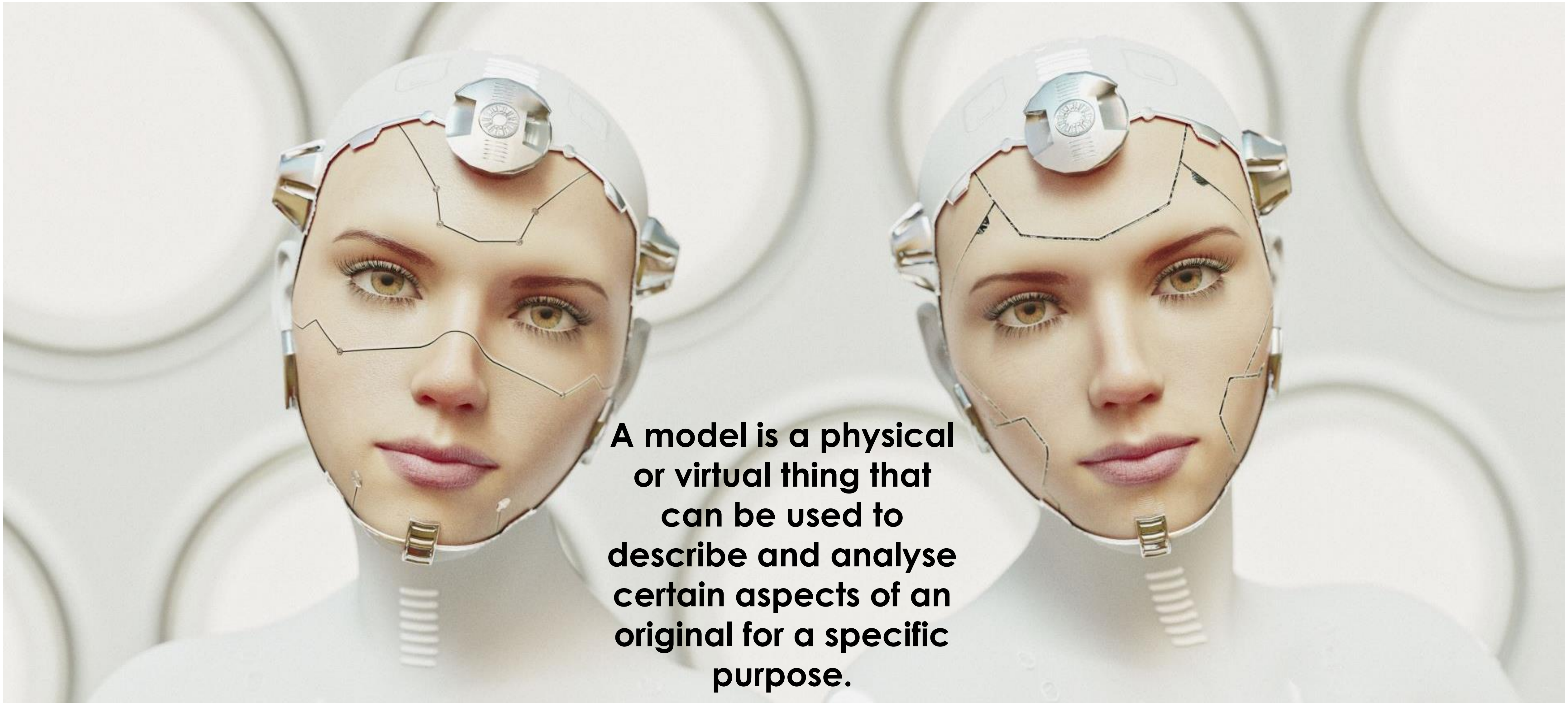
Is built from many sub-systems…

- artificial as well as of natural sub-systems (devices versus organisms, particularly humans)
- developed independently of each other and by different organisations at different times
- groups of sub-systems may be similar, but not identical in terms of structure and/or behaviour
- coupling varies from very tight to very loose and may change over time

The entire OSoS…

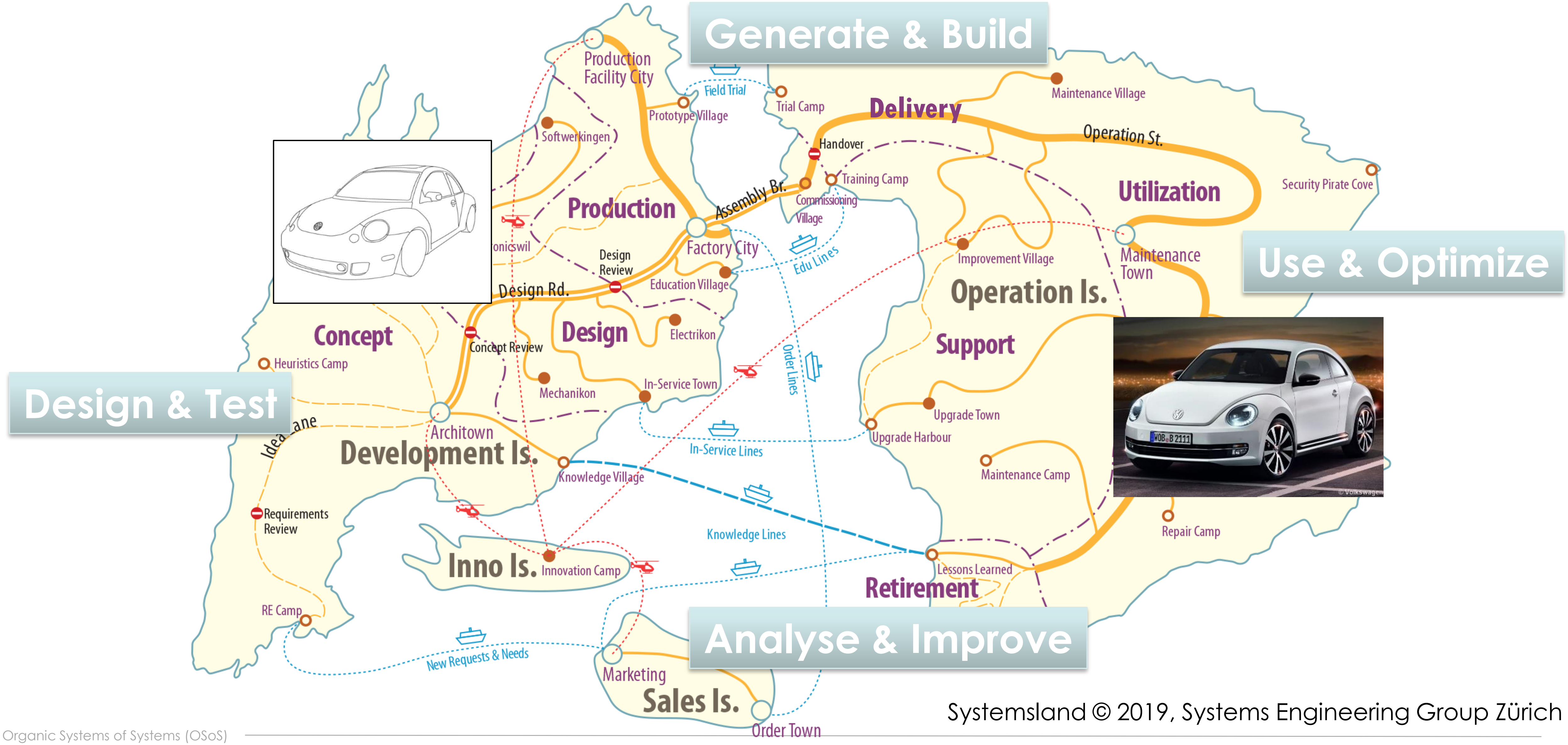
- is resilient and able to adapt to an ever changing, highly dynamic environment
- has a very long life-expectancy (at least decades) with continuous evolution during operation
- may change its anatomy during its lifetime: sub-systems may come and go
- interconnections may change regularly

Models as Digital Twins



A model is a physical or virtual thing that can be used to describe and analyse certain aspects of an original for a specific purpose.

Models enable Simulations



Systemsland © 2019, Systems Engineering Group Zürich

EULYNX

A European initiative in the area of railway signalling, with the aim of reducing the cost and installation time of signalling equipment

- Many components (stationary, moving, human)
- Many organizations (railway operators & industry)
- Long life expectancy (100 years and more)
- Permanent operation and permanent evolution

EULYNX Approach & Challenges

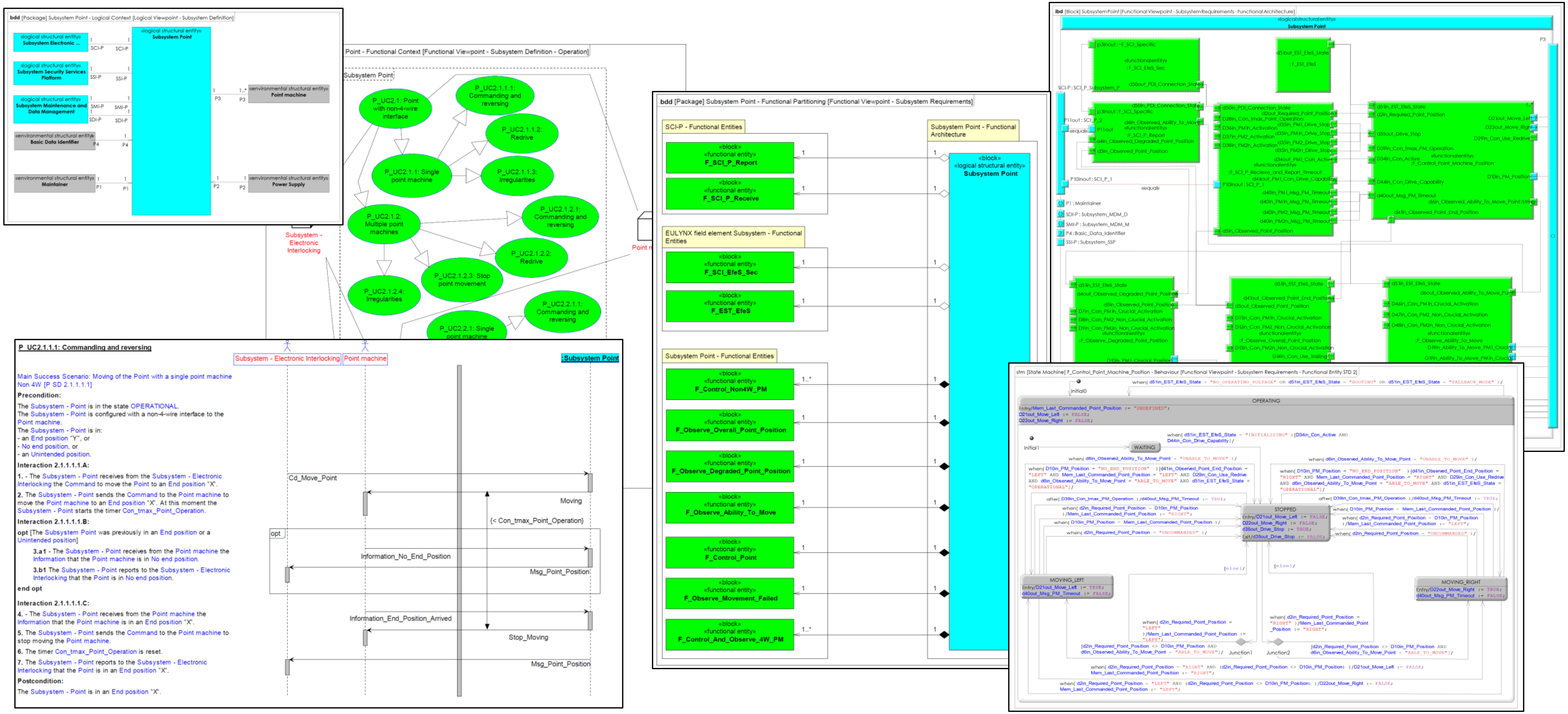
Approach

- Overall architecture is specified using **informal text & graphics** as well as **Arcadia/Capella**
- Component (control) functionality is specified using **executable SysML** (SySim-simulation)
- Operational processes **individually specified** (UML sequence & activity diagrams, rules, Capella, BPMN, Petri Nets, structured text…)
- Physical specifications/simulations based on **Matlab/Simulink** planned for the future

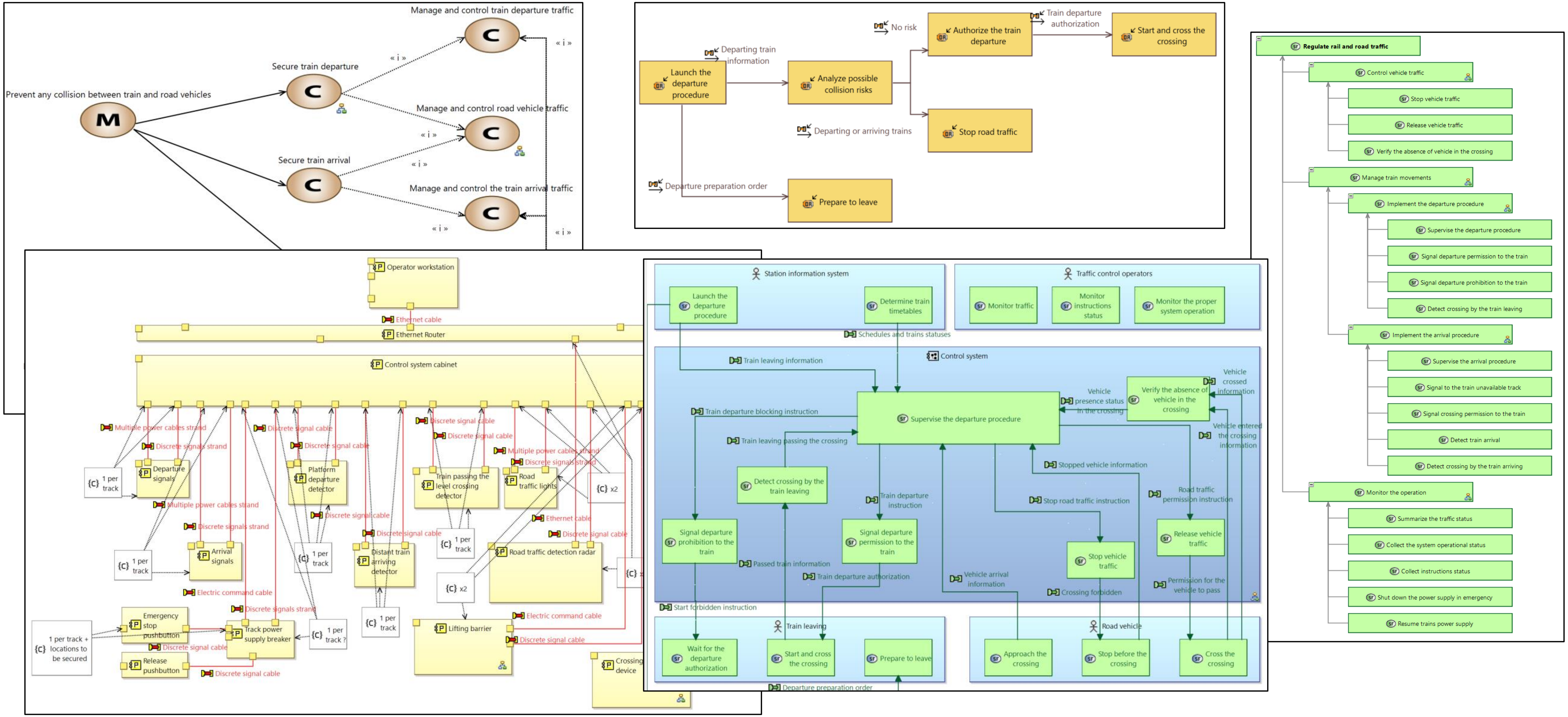
Challenges

- **Integration of modelling approaches** on architecture a functional levels
- **Limitations of SysML/SySim**
  - **"Hard-wired" composition** prevents co-simulation of more than one (!) component
  - **Simplistic action & expression language** ("almost programming")
  - **Missing quantifiers** and dynamic instantiation hinder generic models
- **No formal model-based testing** and thus no automatic regression testing
- **No physical specifications & simulations** (e.g. distributed power management)

EULYNX' SysML Approach



EULYNX' Parent Project: Arcadia Approach



Requirements on Modeling Formalism for OSoS

(Language) Properties

1. **Multi-disciplinary Models:** Description of mechanical, electrical, chemical, communicative, economic, risks, cognitive behaviour, etc.
2. **Precise Models:** Formal enough to support complex analyses as well as simulation scenarios
3. **Dynamic Structure:** Models changing their composition, structure, and configuration at simulation/lifetime while running a simulation
4. **Long-lasting Models:** Based on (minimal) standards so that they may be migrated

(Tool) Capabilities

1. **Model Creation:** Manual and (semi-) automatic creation and update of models
2. **Model Analysis:** Formal checking and proving of model properties and behaviour
3. **Model Execution:** Automatic derivation of model properties and dynamic execution engine
4. **Model Integration:** Black-box and glass box integration of foreign models – even when expressed in "foreign" formalism
5. **Model Transformation:** Generation of the "original" of the model

Modeling Formalisms

- **SysML:** A Systems Engineering modeling language; focused on expressing architecture and functional apportionment of interdisciplinary technical systems; limited behaviour (simulation) and almost no structural dynamics
- **(Executable) UML:** A Software Engineering modeling language; xUML focused in simulation of complex behaviour of any type of objects (not just software; supports generalization/specialization
- Many other **OMG languages** such as BMM (goals and strategies), SBVR (terminology and rules, BPMN (processes), UTP (testing), RAAML (risk), …
- **BIM:** A Civil Engineering language; focused on interdisciplinary design, analysis, simulation and generation of building aspects
- **Modelica & FMI:** Open-source (a)causal modelling language for cyber-physical systems; FMI as an exchange standard for dynamic simulations
- **Other formalisms:** FEM, CFD or proprietary vendor tools such as Matlab/Simulink, Dassault/Dymola, and others

new!

There is no "right" solution!

Improvements of SysML v2 over SysML v1

- Much cleaner system **decomposition semantics**
  - type/instance/use approach for parts, interfaces, …
  - specialization via sub-classification, sub-setting and redefinition (by use)
- **Features** as first-class model elements with constraints
- Improved **action/expression/control language** with more types
- Built-in **variability language** to design product families
- Some support for **test specifications** => UTP
- Some support for **risk modelling** => RAAML
- Based on **KerML** instead of UML

However…

- Still no support for dynamic structures (instantiation of parts, associations, functions, etc.)
- Limited execution semantics
- Does not fit for everything

Summary and further information

- **OSoS** are a type of **long-lasting systems that have a very complex and dynamic structure** and behaviour that is continuously evolving
- A "one size fits all" approach to design and analyse this kind of systems doesn't work: We needed an **integrated federation of modelling formalisms** to design, analyse and simulate such systems even while they are in operation
- **SysML v2** is a substantial improvement over v1, but **still lacks support for dynamic instantiation and structure**

Questions?

Come and join the SE-Group Zürich!

Links

- EULYNX: https://eulynx.eu/
- SysML v2:
  - https://github.com/Systems-Modeling/SysML-v2-Release
  - https://www.knowgravity.com/the-limits-of-sysml-v1-and-how-sysml-v2-addresses-them-2
- Arcadia: https://www.eclipse.org/capella/arcadia.html
- Model Federations: https://www.youtube.com/watch?v=rkxzvTHfgRo