TRIZ • DDCS • ARCHITECTURE

A TRIZ-Based Methodology for Analysis and Architecture Design of Data-Driven Control Systems

Anton Tyukov, EMBA

CTO at imby (Belgium)

Co-author: Adriaan Brebels

Energy Management & Energy Hubs Data & AI

---

01 **Functional analysis**

DFD and function descriptions

02 **Defect identification**

A catalogue of 16 entries

03 **Cause–effect analysis**

Causes and remedies

Anton Tyukov



CTO at imby (Belgium)

From strategy to technology that works

Energy Management & Energy Hubs

Data & AI

EMBA

Synergy University

Contact: Anton.tyukov@gmail.com

The energy transition is creating energy hubs

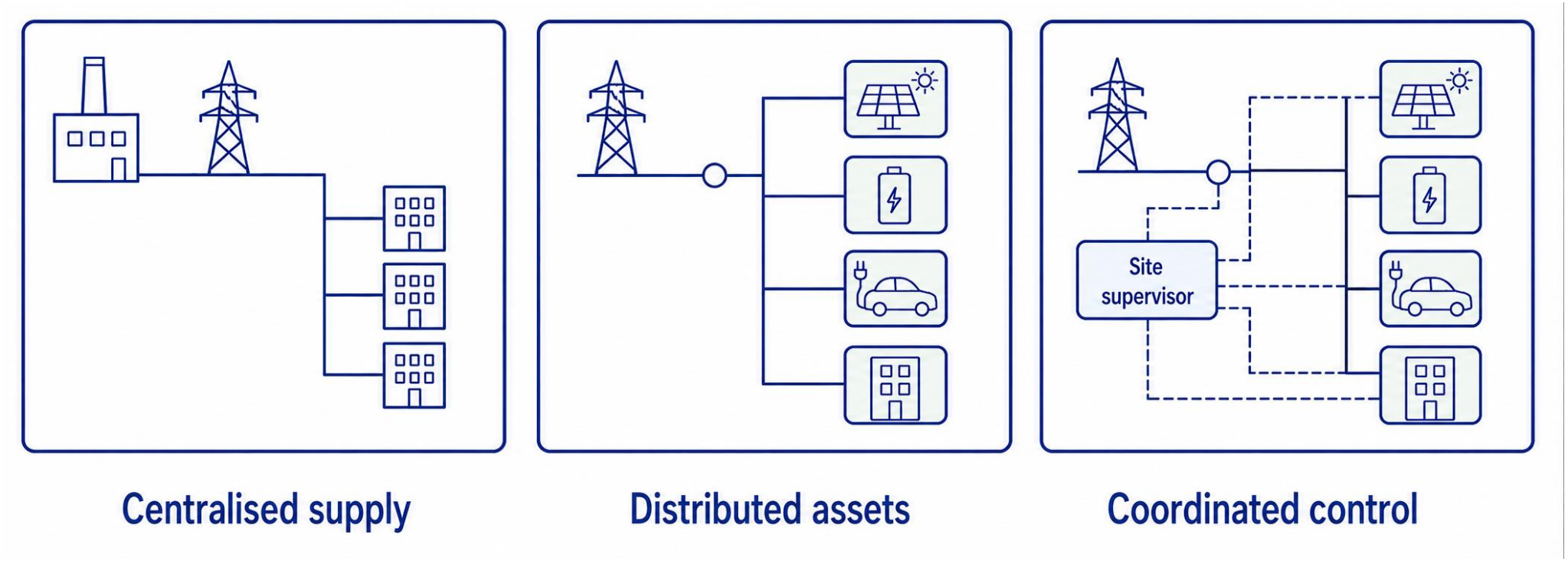
Worldwide in 2025

692 GW

renewable capacity added

85.6%

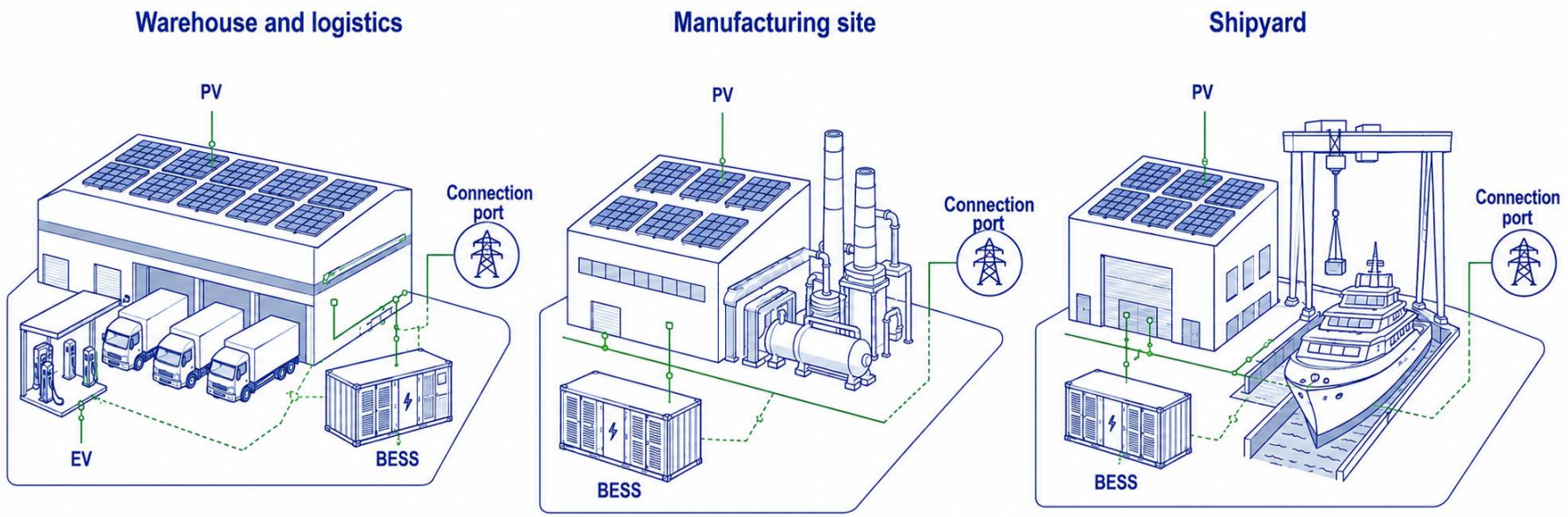
renewables' share of total capacity additions



Renewable capacity increases the need for flexibility. A hub coordinates generation, storage and controllable demand.

Industrial sites within an energy hub

The hub coordinates generation, storage and controllable loads across several industrial sites.



1 MW **5 MW** **10 MW**

Illustrative maximum site capacities, not customer measurements.

Import and export limits are defined separately for each port. Power (MW) and stored energy (MWh) are distinct quantities.

Research problem, object and subject

Research problem	Coordinate the selection of control laws and assignments across horizons as equipment composition, data quality and available functions change.
Research object	Energy and power control processes in energy hubs with heterogeneous equipment and multiple sites.
Research subject	Models of admissible configurations and algorithms for supervisory control, state estimation, diagnosis and functional recovery.

Research goal and objectives

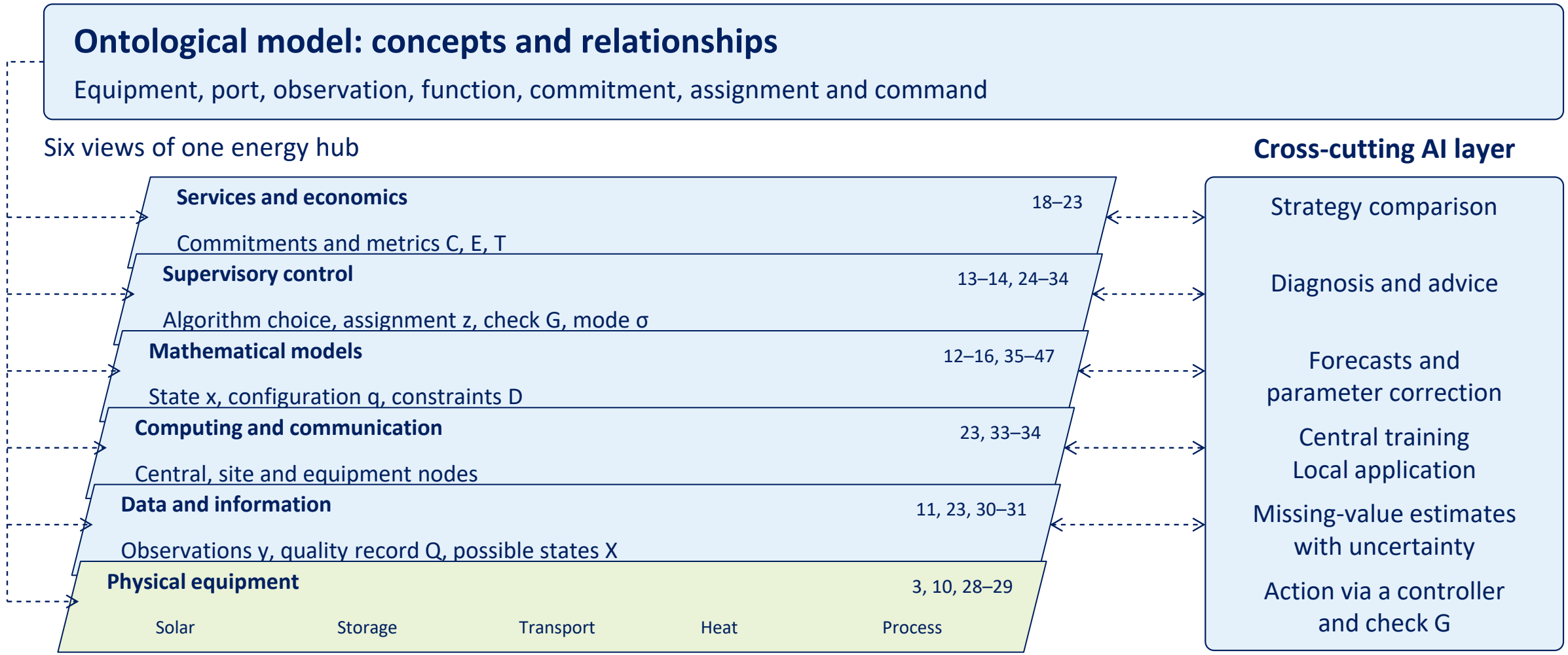
Research goal

Develop and experimentally substantiate the theoretical foundations, models, methods and algorithms for multi-horizon supervisory control of an energy hub with a variable configuration. Preserve system invariants and enable economical, safe operation as load, data availability and external conditions change.

Research objectives

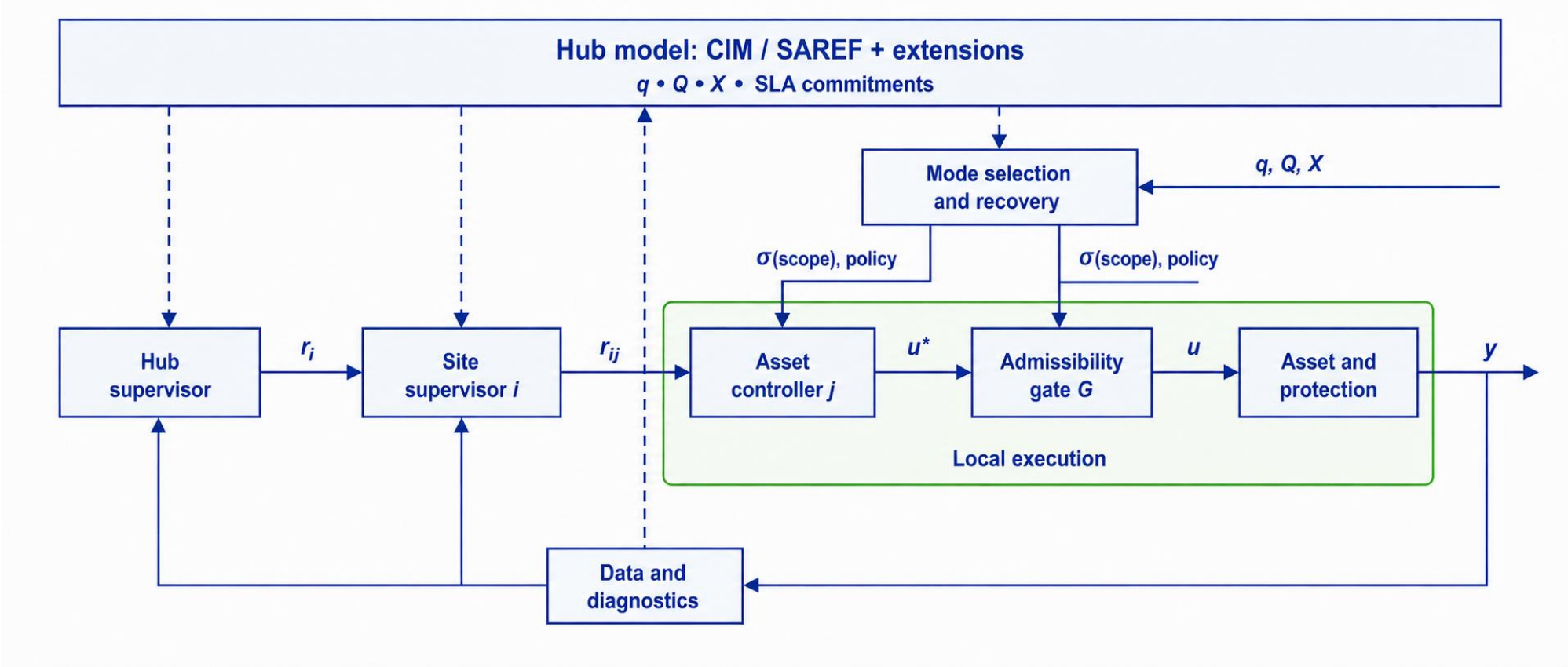
1. A formal hub model, admissible configurations and invariants
2. Local objectives, a shared objective function and constraints
3. Algorithms to coordinate decisions across four time horizons
4. An information model and data admissibility rules
5. Verifiable causal diagnosis of model–observation discrepancies
6. Selection of primary and fallback algorithms; checking changes in q
7. Comparable experimental validation on heterogeneous systems

Energy hub model map



Ontology links the views (11). Supervisor algorithm and admissibility: 13–14. Hierarchy: 17; equipment loop: 29. AI validation and application: 34.

Supervisory control: overall architecture



$x(k+1) = f_q(x(k), u(k), d(k)), \quad y(k) = h_q(x(k)) + v(k)$

σ(scope): the policy and permitted actions for a hub, site, service or asset.

Extending the 2013 supervisory control approach: from building climate control to N sites and repeated admissibility checks.

Site-level control services

Service	Control action and metric	Mandatory conditions
Peak limiting	Battery and load shifting: jpeak = max Pimport	Port limit and agreed peak limit
PV and storage	Charge, discharge, PV curtailment: jPV = Σ(Pavail − PPV)Δt	SoC, wear, export limit
Schedule tracking	Power allocation: jtrack = Σ|Pport − r|Δt	Valid setpoint and resource reserve
Thermal processes	Heating, ventilation, cooling: energy and cost	Temperature, humidity and process quality
Cooling and processes	Refrigeration, pumps, compressors: energy and cost	Product temperature, pressure, workload
EV charging	Charging schedule: jEV = Σ(Ereq − Edeparture)$_+$	Energy by deadline and charger-group limit

Service metrics define conditions for the overall problem. Energy and wear enter C once.

Process requirements take priority: product quality, pressure, temperature and the production schedule.

Services across hub sites

Service	Hub supervisor decision	Delivery conditions
Energy sharing	Coordinating import, export and storage schedules $E_i(t)$	Physical topology, energy metering, contracts and accepted nominations
Power sharing	Allocating available limits $P_i(t)$ and reserve across sites	Each port's limits, rights to shared resources and allocation validity
Joint response	Allocating response and replacing unavailable resources	Site availability, shared commitments and no double reservation

Energy is measured in kWh, power and reserve in kW. Setpoints r_i link hub services to site control.

Coordination respects physical connections and contractual rights. A virtual hub alone creates neither a physical link nor net settlement.

Why architectural analysis is needed

A complex software–hardware ecosystem

Technical debt

Component duplication

Architectural contradictions

PROBLEM STATEMENT IN THE ARTICLE

From predominantly reactive analysis
to a systematic search for architectural defects.

Code review • Architectural Decision Records • ISO/IEC 25010

Research aim and four objectives

Develop and validate a methodology for transferring TRIZ tools to DDCS architecture design.

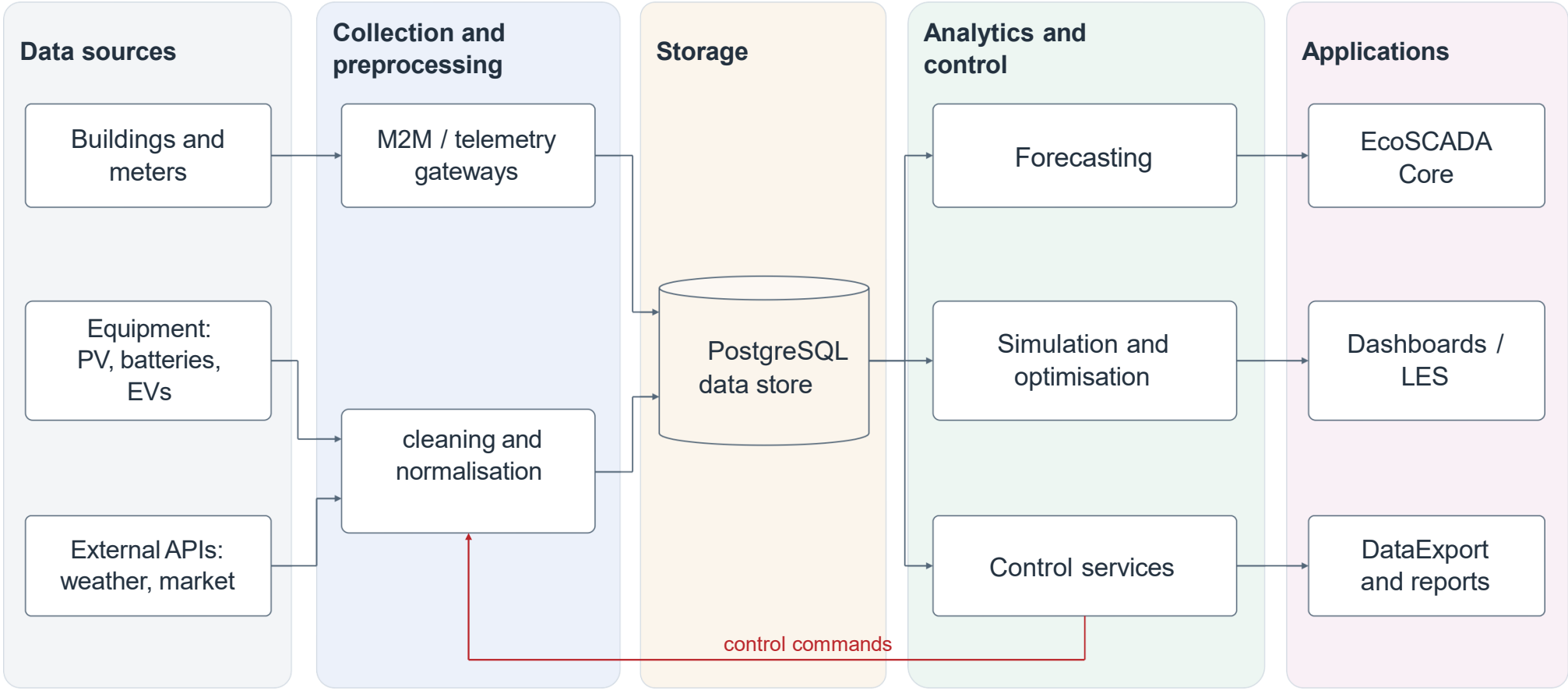
01 Substantiate the structural analogy between Miles and DFD.

02 Develop a defect catalogue classified using seven alphas.

03 Propose remedies for each defect.

04 Present the results of the case study.

Conceptual architecture (IFR)



DFD captures processes, data flows and stores; this structure provides the input for TRIZ-based architecture analysis.

Source: § 2.1; Figure 1

The Functional analisys –DFD structural analogy

FUNCTIONAL ANALYSIS / MILES

$$FA: \langle \text{Subject} \rightarrow \text{Function} \rightarrow \text{Object} \rangle \quad (1)$$

DATA FLOW DIAGRAM / DFD

$$DFD: \langle \text{Process} \rightarrow \text{Data flow} \rightarrow \text{Data store} / \ \text{Terminator} \rangle \quad (2)$$

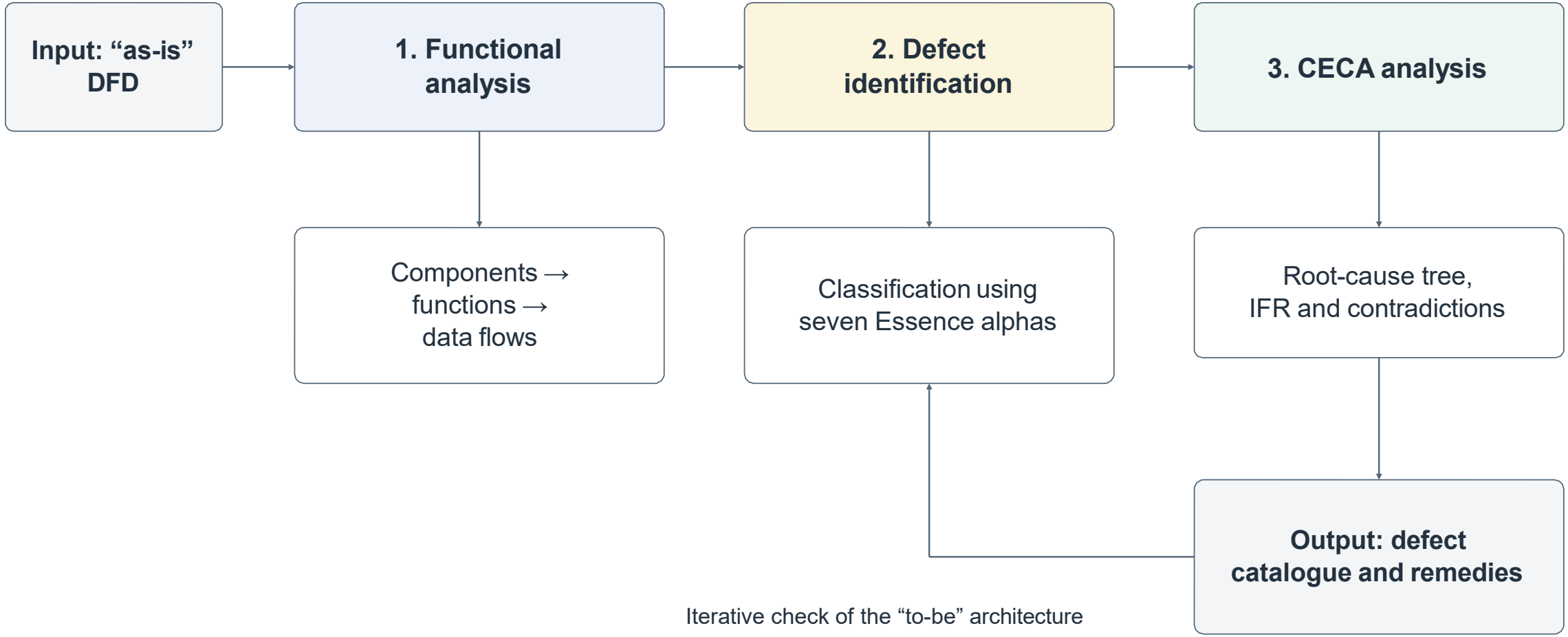
MAPPING φ

Subject ↦ Process	**Function ↦ Data flow**	**Object ↦ Data store**

(3)

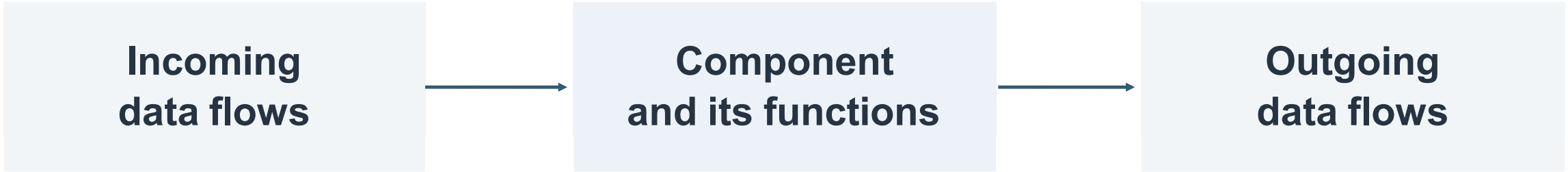
The article calls this mapping a formal isomorphism φ and uses it as the basis for transferring TRIZ tools.

The three-stage TRIZ analysis methodology



Stage 1. Functional analysis

Functions and data flows are identified for each component



Overview DFD

A system description for non-technical stakeholders

Detailed diagrams -> Tables

Navigation to functions and relationships at individual levels

Stage 2. Identifying generalised defects

Systematic review of the “as-is” diagrams

Stakeholders	**Opportunities**	**System definition**	**System realisation**
Team	**Work**	**Technologies**	

Methodological basis

An analogy with Lenyashin’s 30 generalised defects

The seven alpha names follow the terminology used in the article.

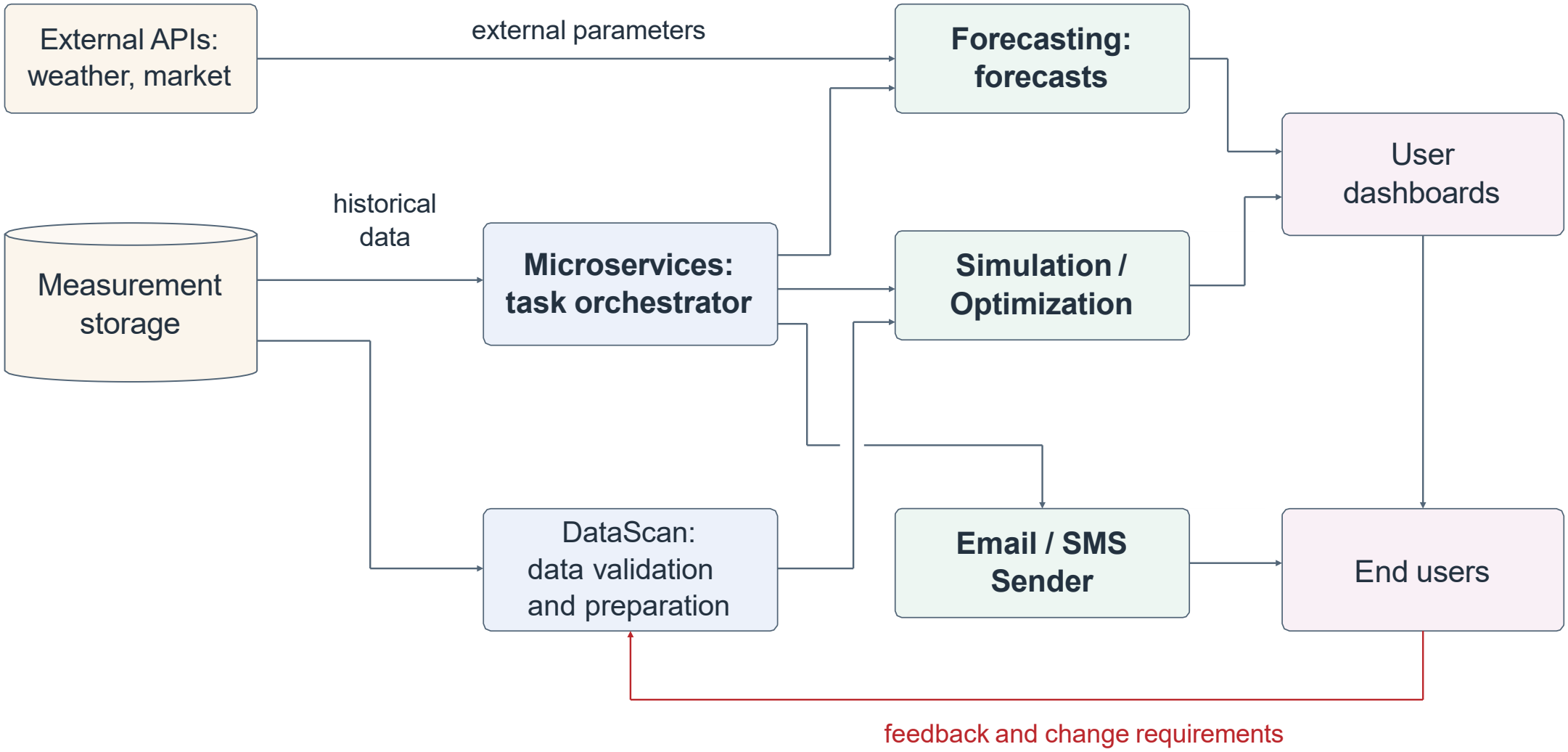
Stage 3. Cause–effect analysis

From a defect to its causes and possible remedies

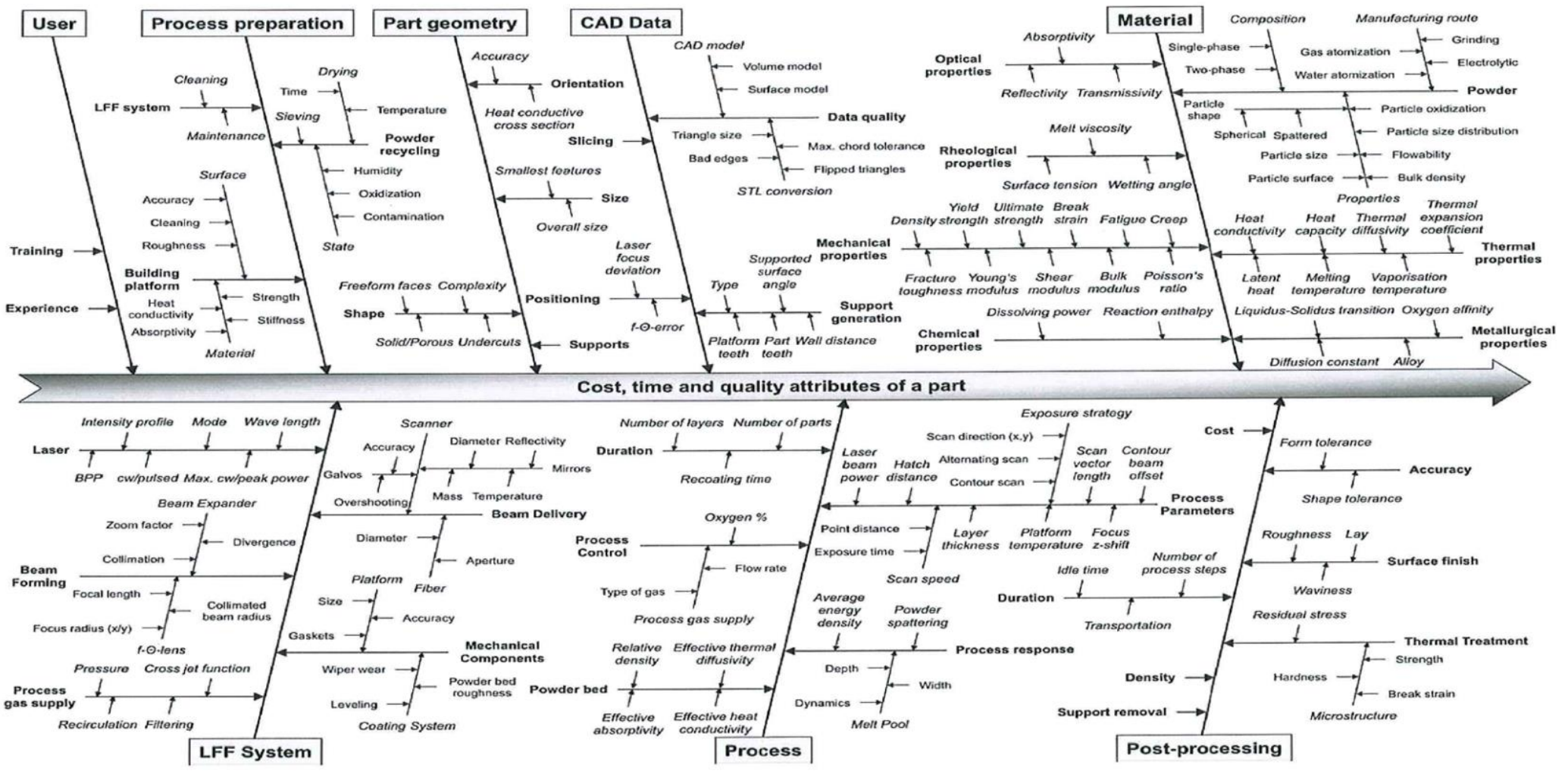


IFR — Ideal Final Result

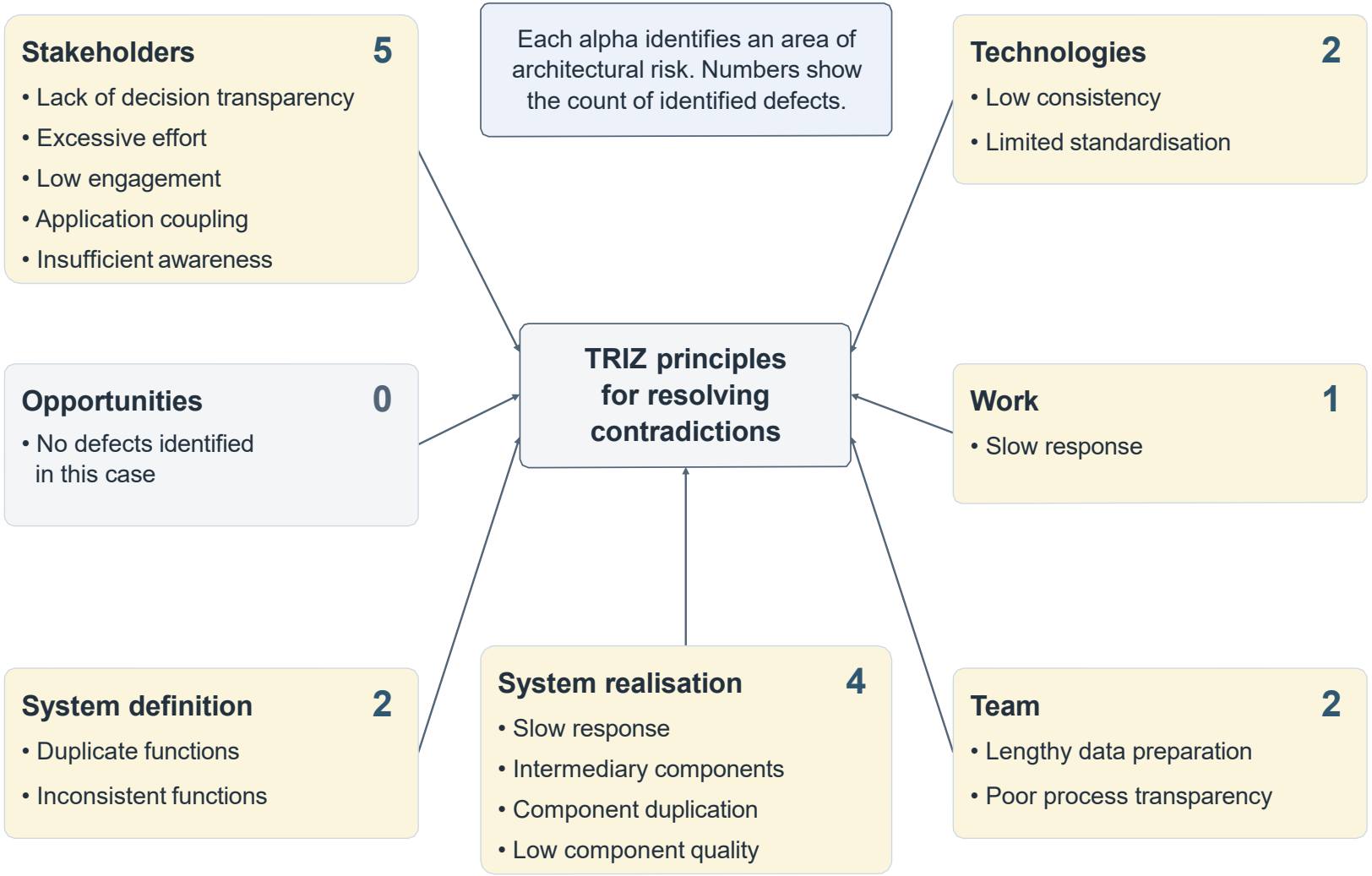
DFD of the analytical services



Ishizawa diagram



Map of generalised architectural defects (To delegate responsibility)



16

generalised defects

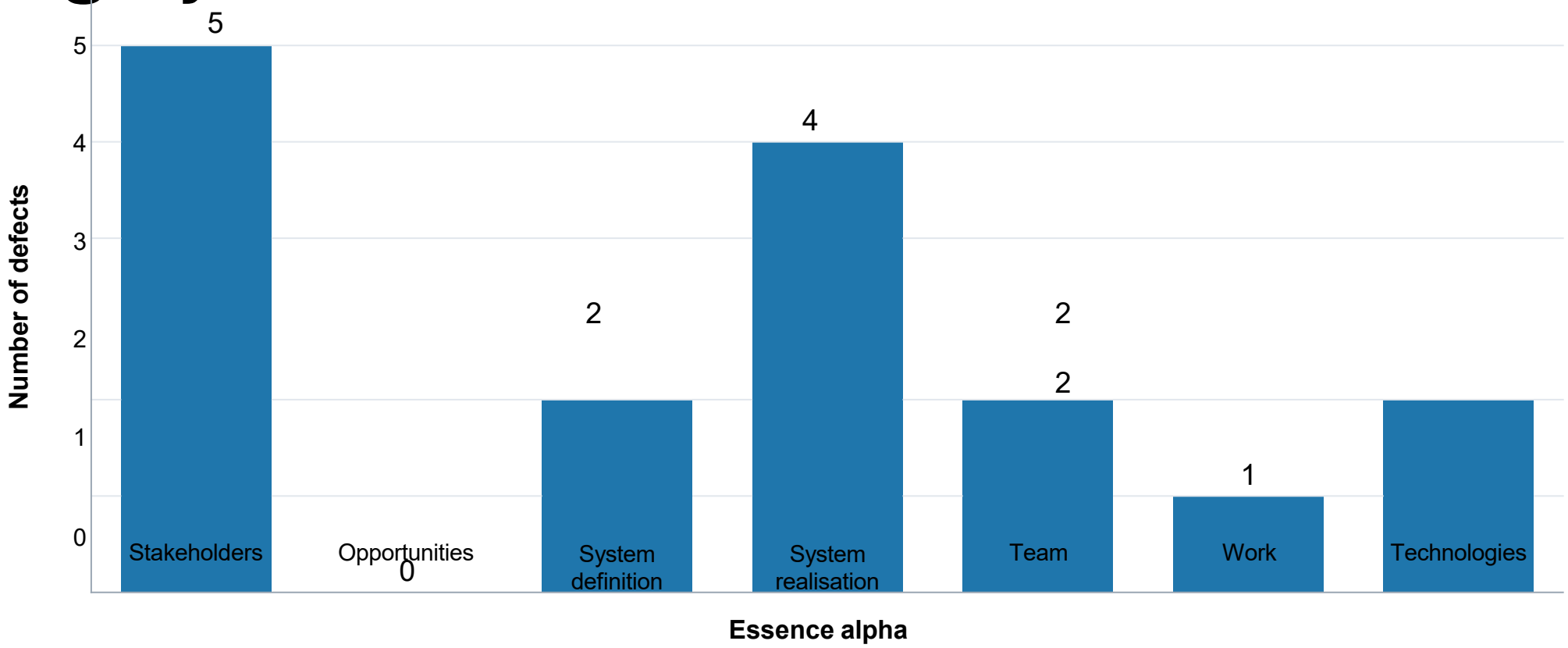
7

categories in the article's classification

Full breakdown: slides 16–23

Source: § 3.2; Figure 4; Table 1

Distribution of the 16 defects by category



In this case, the largest groups of problems concern stakeholders and system realisation.

Source: § 3.2; Figure 5

System realisation: transmission and processing

DEFECT	CAUSE	REMEDY
Slow data transmission	Too many intermediary components; low throughput	Remove intermediaries; transfer functions to the supersystem; use caching
Intermediary data-processing components	Uncoordinated development work	Coordinate activities using a shared architecture diagram

System realisation · extract from the 16-defect catalogue

System realisation: duplication and quality

DEFECT	CAUSE	REMEDY
Component duplication	Several components perform the same function	Coordinate components; discontinue support for the duplicate component
Inadequate operational quality of components	No evaluation criteria	Develop a tool for assessing operational quality

System realisation · extract from the 16-defect catalogue

Technologies: consistency and standardisation

DEFECT	CAUSE	REMEDY
Low consistency of technologies	The same functionality is implemented by different components	Create an Architecture Definition Document (ADD)
Limited software standardisation	Applications of the same type use different programming languages	Create an ADD and a technology-stack standard

Technologies · extract from the 16-defect catalogue

System definition: application functions

DEFECT	CAUSE	REMEDY
Duplicate functions across applications	Several applications repeat the same functionality	Redistribute functions; combine them into a functional group
Inconsistent functions across the ecosystem	Users need several applications to access the full functionality	Redistribute functions into functional groups

System definition · extract from the 16-defect catalogue

Stakeholders: decisions, information and engagement

DEFECT	CAUSE	REMEDY
Lack of transparency in decision-making	The application displays data but does not explain the action	Add an explanation module; move towards automatic control
Excessive effort to obtain information	Too much effort is needed to obtain data in time	Push delivery to mobile phones and email
Low user engagement	Users do not understand their own objectives in the system	Motivation system; gamification; a feedback loop

Stakeholders · extract from the 16-defect catalogue

Stakeholders: coupling and awareness

DEFECT	CAUSE	REMEDY
Excessive coupling between applications	Applications call one another unnecessarily	Refactoring; loosely coupled architecture
Insufficient management awareness of the architecture	Rapid changes; poor-quality descriptions	Component description standards; regular architecture reviews

Stakeholders · extract from the 16-defect catalogue

Source: § 3.2; Table 1, data rows 12–13

Team: data preparation and process monitoring

DEFECT	CAUSE	REMEDY
Long data preparation before analysis	Non-standardised input data formats	Standardise formats; automate data preprocessing
Lack of transparency in technical-process monitoring	It is unclear which components are running	Link client, server, data and data quality for monitoring

Team · extract from the 16-defect catalogue

Work: responding to change requests

DEFECT	CAUSE	REMEDY
Slow response to change requests	Slow management processes; lack of standards	Improve awareness; maintain standards; shorten the cycle

Proposed direction: improve awareness, maintain standards and shorten the cycle.

Source: § 3.2; Table 1, data row 16

Practical value of the methodology

Proactive analysis

Defects are identified before an incident occurs.

Shared communication

Generalised descriptions give a distributed team a common language for discussing problems.

Contradictions and the IFR

Define the “to-be” state of the architecture within the proposed methodology.

Scientific results as stated in the article

RESULT
Formulating architecture analysis through a system of TRIZ printiples.
Assessing architecture decisions using the defect catalogue and seven Essence alphas.
Algorithmic support for the three-stage TRIZ analysis.
Problem-oriented design of control-system architectures.

Item numbers and result statements follow the article’s conclusions.

Limitations and scope of application

01 **One industrial case**

The catalogue was developed from the experience of one company.

02 **One class of systems**

Energy-flow management.

03 **Conditions for application**

The team requires a specialist with at least MATRIZ Level 1 certification.

ERP, biomedical DDCS and telecommunications require separate validation.

Directions for further research

Extending the catalogue

Towards equivalents of 30 IT defects
and 40 IT inventive principles.

Other classes of DDCS

Validate the methodology
for new classes of systems.

OTSM–TRIZ

An ontology of
architecture development.

Automated reviews

Tool support for
TRIZ-based architecture analysis.

These are future research directions, not a list of implemented functions.

How triz helps me as for CTO

- TRIZ strengthens my critical thinking and helps me challenge assumptions.
- TRIZ is particularly valuable in complex system development, where small changes can have far-reaching consequences.
- Formalizing IT design principles helps focus attention on critical system properties and architectural decisions.
- TRIZ helps me decompose processes into clearly defined tasks that I can delegate to people or AI agents.
- Converting diagrams into structured tables makes system knowledge easier for AI tools to interpret and process.
- AI accelerates analysis and execution, while creative thinking remains essential for framing problems and developing original solutions.

From an architectural model to directions for improvement

Miles–DFD

Mapping notations as the basis for transferring the tools

Three stages

Functional analysis → defect identification → CECA

16 defects

An EcoSCADA catalogue with causes and remedies

Anton P. Tyukov · Adriaan Brebels

Anton.tyukov@gmail.com

Acknowledgements Ts2RED-AI, Volgograd State Technical University · Agreement No. 70-2025-000756
Yu. Danilovsky — methodological materials on functional analysis.

References cited in the article · 1/2

[1] OMG Essence Standard — Kernel and Language for Software Engineering Methods, v1.2. Object Management Group, 2018. 182 p.

[2] Altshuller G.S. Algorithm of Invention. 2nd ed. Moscow: Moskovsky Rabochy, 1973. 296 p. (In Russian).

[3] The Open Group. TOGAF Version 9.2. Van Haren Publishing, 2018. 496 p.

[4] ISO/IEC 25010:2011. Systems and software quality models. Geneva: ISO, 2011.

[5] Altshuller G.S. Creativity as an Exact Science. Gordon and Breach Science Publishers, 1988. 320 p.

[6] Dvorak J., Busov B. Use of TRIZ Tools in Software Engineering. Proc. 10th ETRIA World TRIZ Future Conference. 2010. pp. 109–118.

References cited in the article · 2/2

[7] Fowler M. Refactoring: Improving the Design of Existing Code. 2nd ed. Addison-Wesley, 2018. 448 p.

[8] Visual Paradigm International. Visual Paradigm Enterprise Edition. https://www.visual-paradigm.com (accessed 1 March 2024).

[9] Danilovsky Yu. Functional Analysis Course in Miles Notation. TRIZ Master training materials. 2016. (In Russian).

[10] Lenyashin V.A. Classification of Technical-System Defects. Proc. International TRIZ Conference. St Petersburg, 2003. (In Russian).

[11] Fey V., Rivin E. Innovation on Demand: New Product Development Using TRIZ. Cambridge University Press, 2005. 248 p.

[12] Bass L., Clements P., Kazman R. Software Architecture in Practice. 3rd ed. Addison-Wesley, 2012. 624 p.