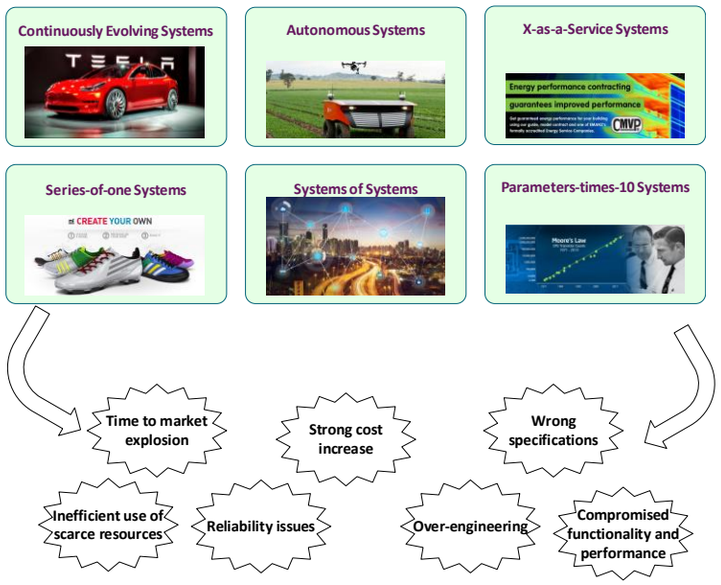
DAARIUS for Transparent Team-based System Design

Trends in complex high-tech systems *Why*



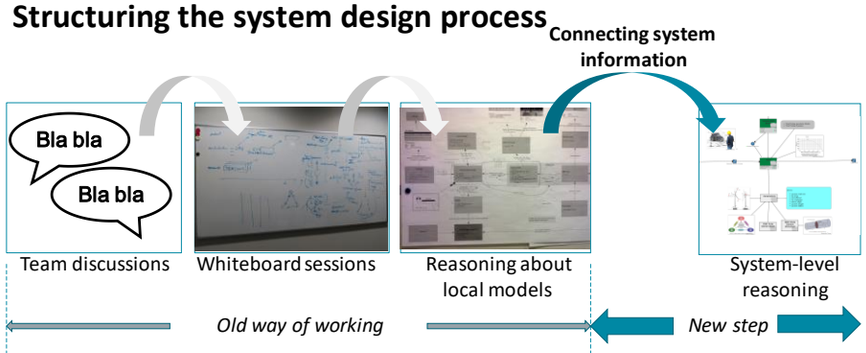
Serious challenges!

- Dealing with the increase in complexity on all levels: business, system, processes and organization.
- Reasoning about system aspects across disciplines and departments.
- Supporting and substantiating business-critical decisions.

Systematic, team-based way of working *How*

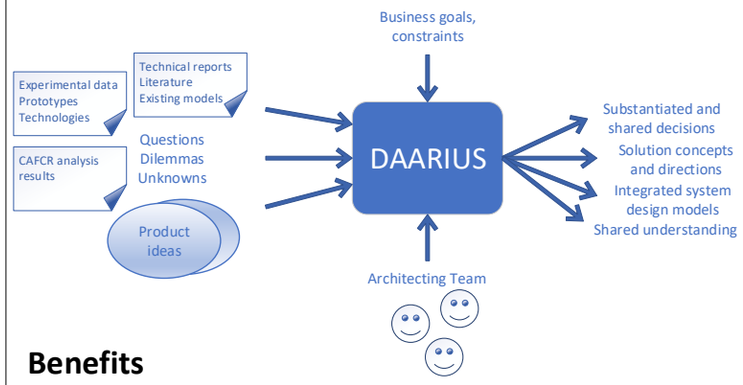
- Capturing and organizing the already existing body of knowledge
- Decomposing the knowledge into a central set of system views, stakeholder views, individual domain views and building blocks views, using step-by-step guidelines
- Making the relations and dependencies between these views explicit using carefully selected key parameters of system and context
- Leveraging the use of simple, executable models for analysis, making trade-offs and decision taking.
- Clearly communicating and collaborating in the system design team across disciplines
- Defining explicit ownership

Structuring the system design process



What

DAARIUS is a structured, scalable, and team-based system design methodology providing traceable underpinning for key design decisions and leveraging the abundance of simple executable models in systems engineering.



Benefits

Provides quickly system overview and insights

- common understanding of what is key and decisive, what are the trade-offs
- obtain clarity of decisions, obtain overview of the context

Enables understanding business and system consequences of system design decisions

- explicit relations enable system level reasoning

Delivers systematic system design

- agreement on critical aspects
- clarity on team progress

Provides a consistent system representation

- single source of truth: consolidated system knowledge
- away from document-based processes

Leads to

improved product quality | **better fit to the market** | **shorter time-to-market**

Digital assistance: DAARIUS tool *Support*

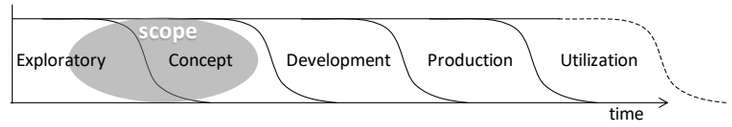
- Representing views, relations, dependencies and parameters in an interactive digital executable environment
- Multi-user, web-based
- Central storage (single source)
- Easy to learn basic concepts, creating modeling freedom
- Interoperability with external modeling tools (Excel, Python, Matlab, Frink, LibreOffice)
- Industry scalable

Training

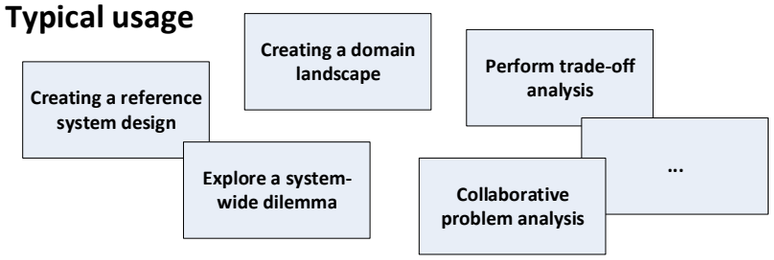
- Workshops
- Manuals/cookbooks

When to apply *When*

- Typically applied in the development of systems/products, when there is limited knowledge, uncertain information and unclarity.
- In (disconnected) multi-disciplinary teams.
- In the early phases of the INCOSE product development lifecycle:

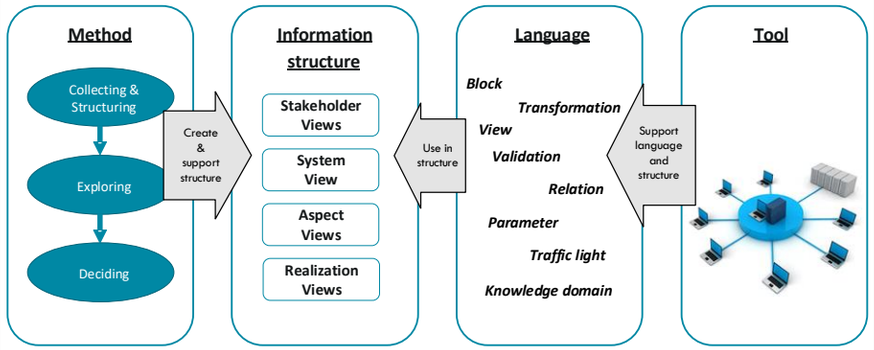


Typical usage

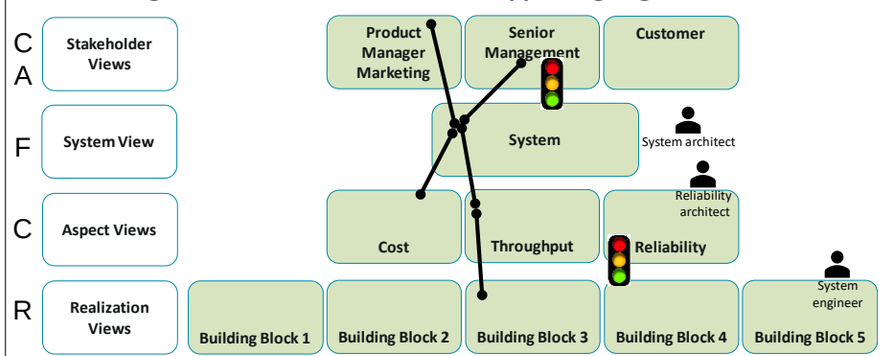


Concepts

Methodology elements:



- **Explicit reasoning across levels and disciplines**
- **The broad view on system information is structured in layers, based on CAFCR [1]**
- **The relations between layers and domains are made explicit**
- **Traffic lights indicate inconsistencies, supporting alignment discussions**



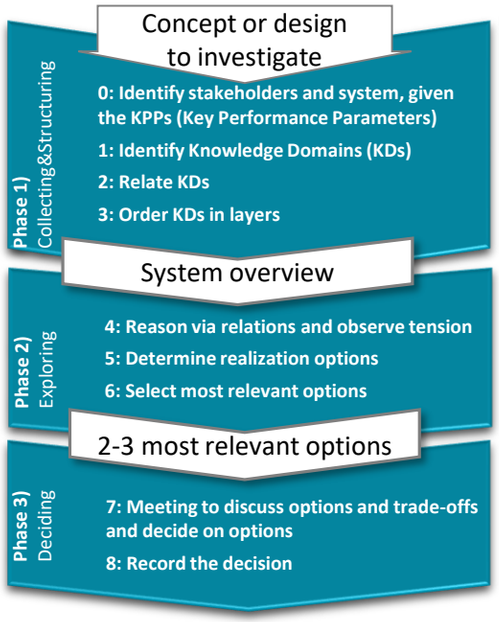
[1] Gerrit Muller, www.gaudisite.nl

DAARIUS methodology: reasoning, communication and decision making for system design

Method

Three phases to setup a reasoning structure and come to a system design decision

- Iterate phases or steps to add details and quantifications
- Reasoning enables tension to be understood, such that options can be chosen

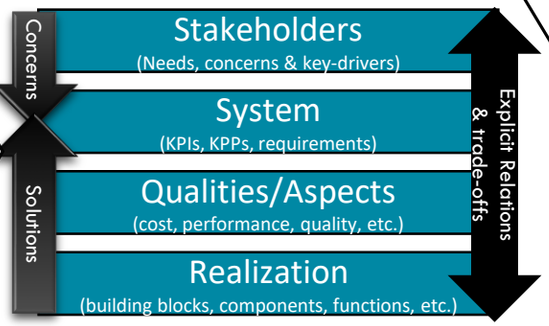


creates & supports structure

Structure

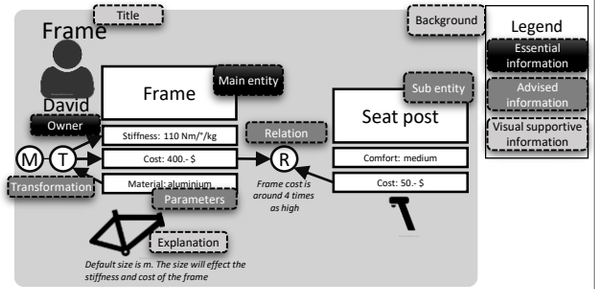
Dashboard provides a top-level system overview[1]

- **Owned by system architect**, which sets targets
- Layers contain and group knowledge domains
- Filter information; show **system level trade-offs**
- **Constraint based design; validations decouple** layers and domains



Knowledge domain views for details[2]

- Makes information explicit and ordered
- Knowledge domain **owned by domain architect**; determine the **scope** and take **decisions**
  - 5 till 7 parameters important for dashboard



used in structure

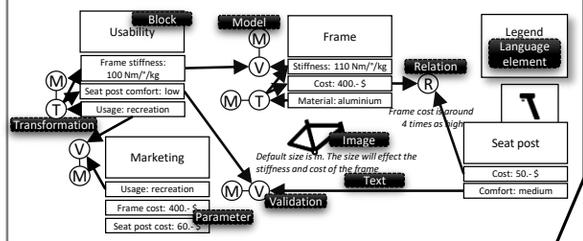
Language

Visual language to capture system knowledge[2]

- **Relations** make trade-offs visible and **traceable**
- Supports **qualitative** and quantitative **reasoning**

Note: quantitative modeling requires 10-100 times more effort!

Example of language elements:



Description of visual language elements:

Language element	Reasoning	Usage
Block	Qualitative	Creates structure, or qualifies an entity, with its name.
Image		Clarifies or illustrates other LEs.
Text		Annotates or explains a LE.
Parameter	Quantitative	Quantifies an aspect from a block.
Model		Makes the relation of a validation or transformation quantitative and explicit. Examples are a mathematical expression, a simulation, executable code, or a loop-up table.
Relation	Qualitative	A qualitative relation between parameters, blocks, images, or text.
Validation	Quantitative	A comparison of parameter values to satisfy a given condition.
Transformation		A transformation of input parameter values into output parameter values.

Supports language and structure

Tool

Web-based tool[3] allows multiple stakeholders to work in parallel on one structure

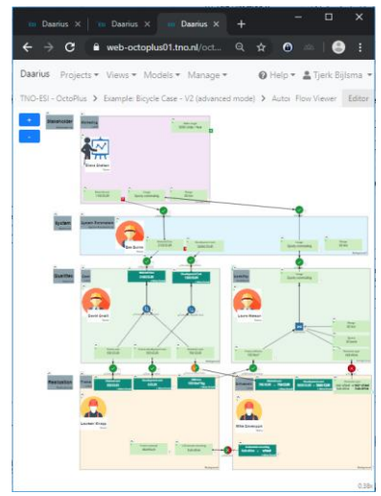
- Digital express/visualize DAARIUS language
- Record history with versions of structure

Visualize a **single structure in multiple views**, instead of loosely coupled documents

- Each view shows a part of underlying structure

Use **executable models** for transformations

- Investigate impact of options by using executable models that compute values for related parameters



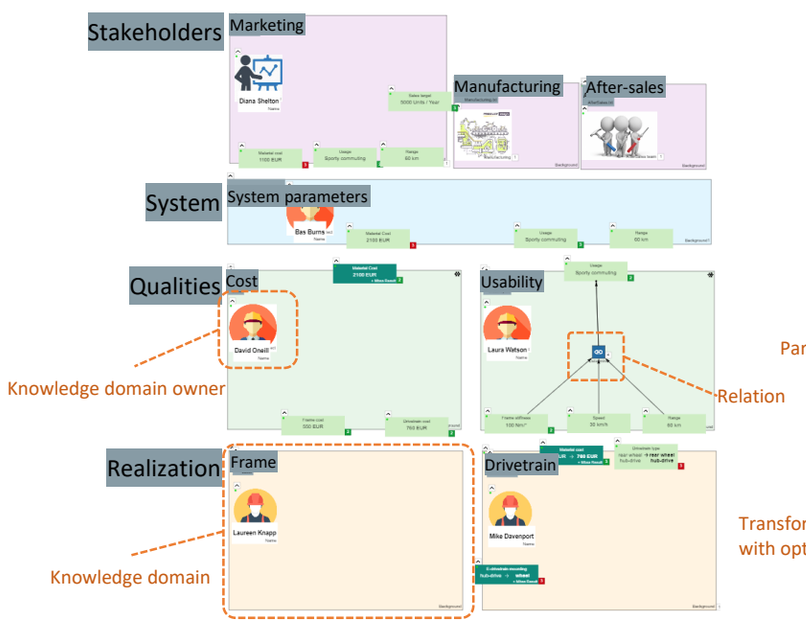
1) T. Bijlsma, B. vd Sanden, Y. Li, R. Janssen, R. Tinsel, "Decision support methodology for evolutionary embedded system design", ISSE 2019

2) T. Bijlsma, W. Tabingh Suermondt, R. Doornbos, "A knowledge domain structure to enable system wide reasoning and decision making", CSER 2019

3) H. Moneva, R. Hamberg, T. Punter "Design Framework for Model-based Development of Complex Systems", RTSS 2011

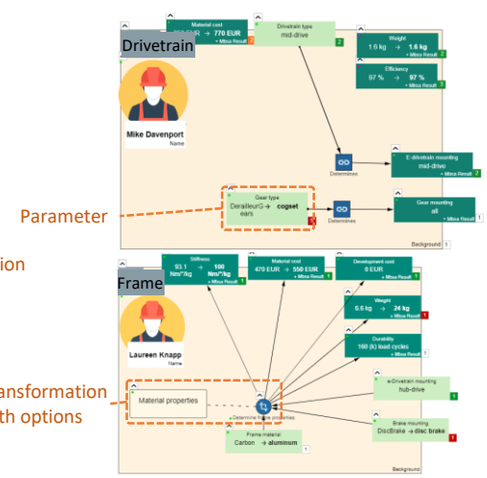
Example: *e-bike system design*

Phase 1) System architect identifies knowledge domains and first relations; Initial dashboard:

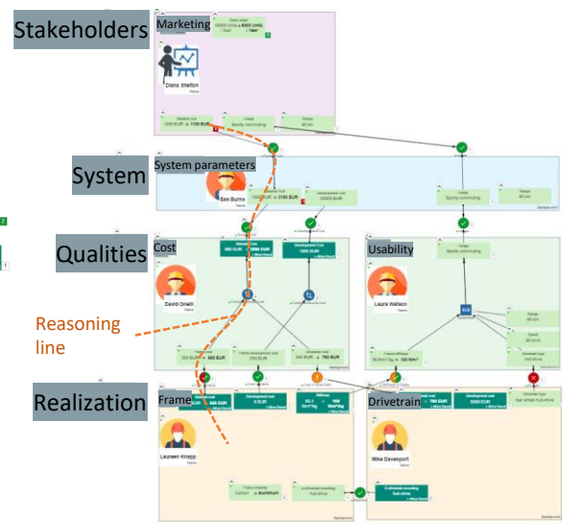


Phase 2) Add information and select interesting options

Domain architect details knowledge domain, adds parameters and makes it possible to select options:



System architected manages dashboard; Reason over options with related knowledge domains:



Phase 3) Architects use dashboard to explain dilemmas & trade-off, and decided