

Module 33, Architectural Reasoning Design Fundamentals

by *Gerrit Muller* HBV-NISE

e-mail: gaudisite@gmail.com

www.gaudisite.nl

Abstract

This module discusses fundamental design methods and techniques, especially partitioning, interface, behavior, and quantified performance design.

Distribution

This article or presentation is written as part of the Gaudí project. The Gaudí project philosophy is to improve by obtaining frequent feedback. Frequent feedback is pursued by an open creation process. This document is published as intermediate or nearly mature version to get feedback. Further distribution is allowed as long as the document remains complete and unchanged.

June 4, 2017

status: preliminary

draft

version: 1.1

System Partitioning Fundamentals

by *Gerrit Muller* Buskerud University College

e-mail: `gaudisite@gmail.com`

`www.gaudisite.nl`

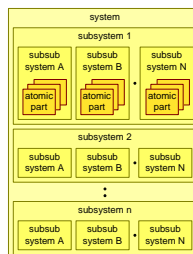
Abstract

The fundamental concepts and approach system partitioning are explained. We look at physical decomposition and functional decomposition in relation to supply chain, lifecycle support, project management, and system specification and design.

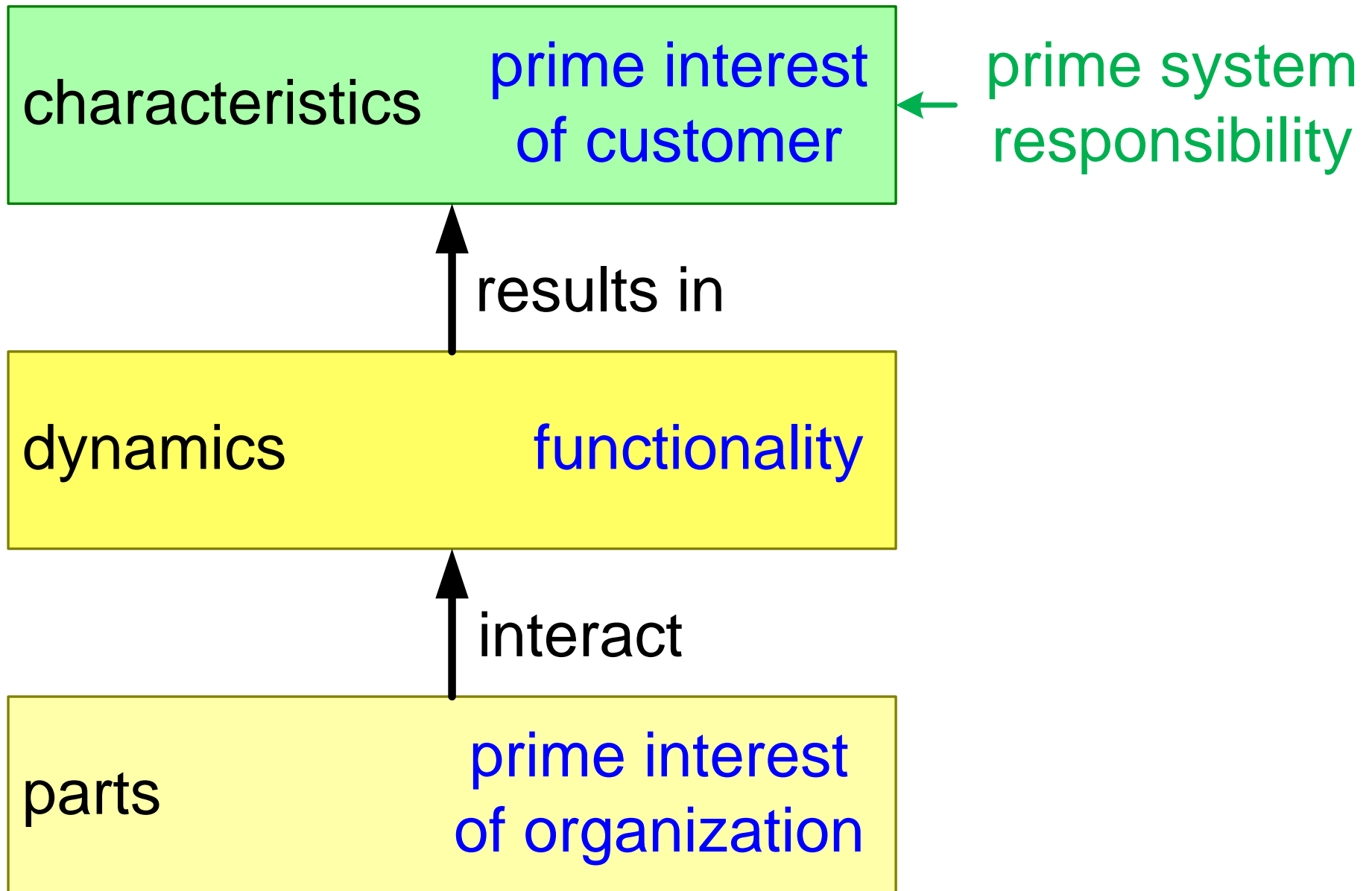
Distribution

This article or presentation is written as part of the Gaudí project. The Gaudí project philosophy is to improve by obtaining frequent feedback. Frequent feedback is pursued by an open creation process. This document is published as intermediate or nearly mature version to get feedback. Further distribution is allowed as long as the document remains complete and unchanged.

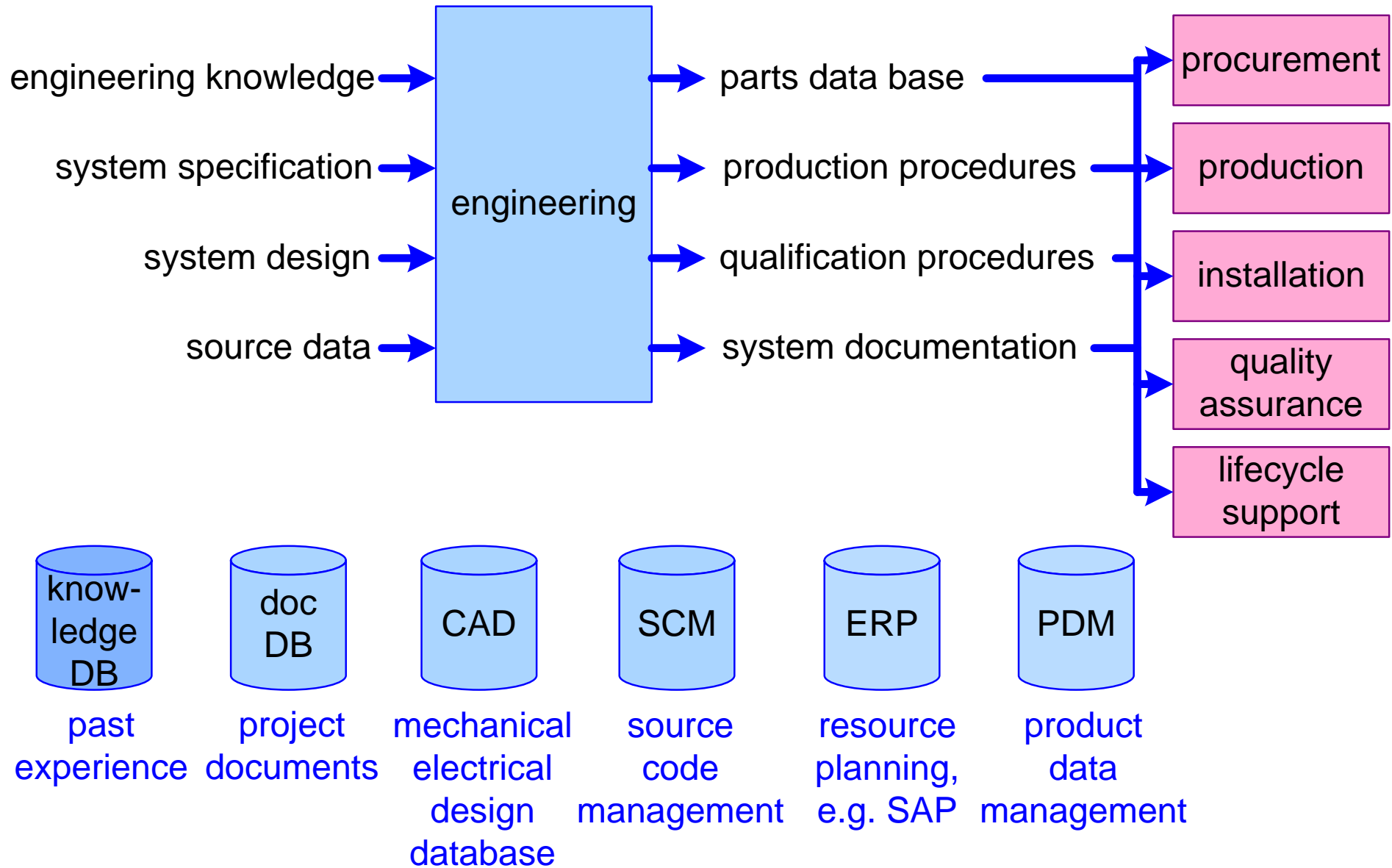
June 4, 2017
status: preliminary
draft
version: 0.2



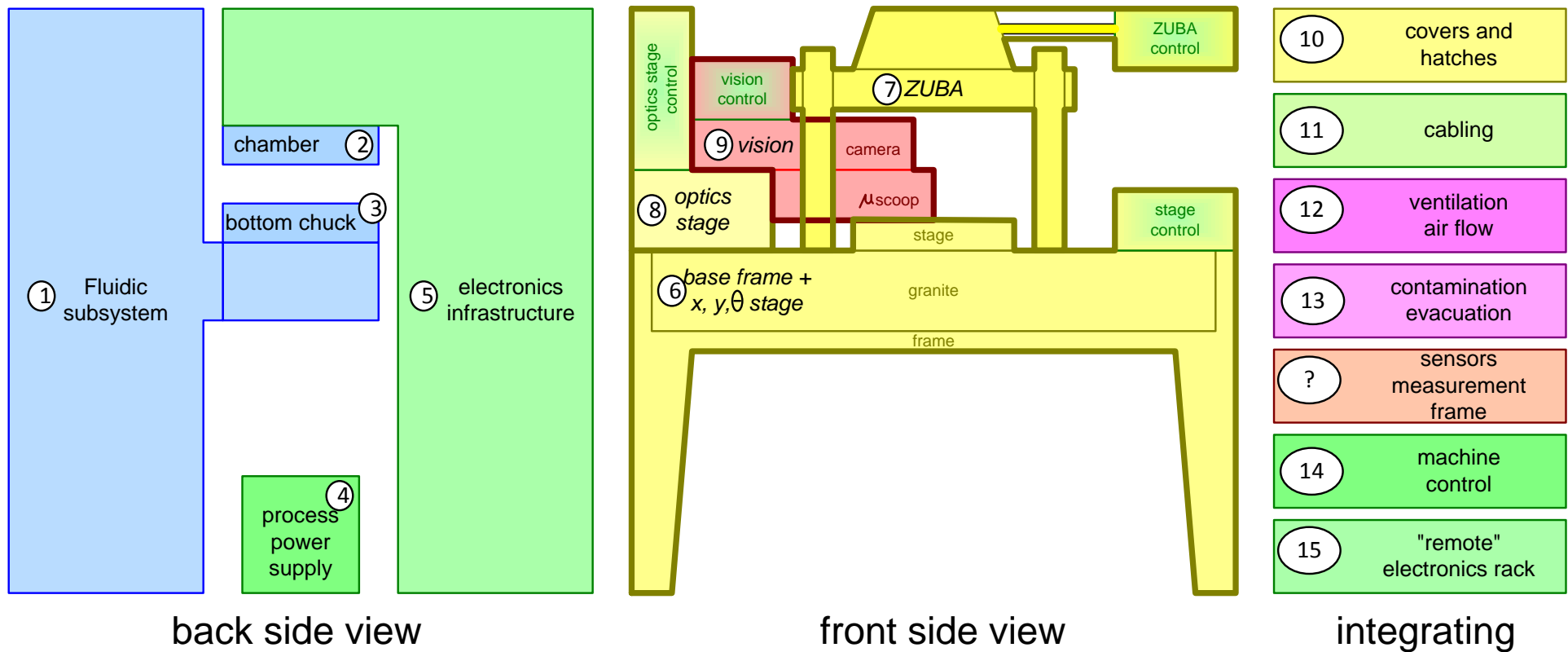
Parts, Dynamics, Characteristics



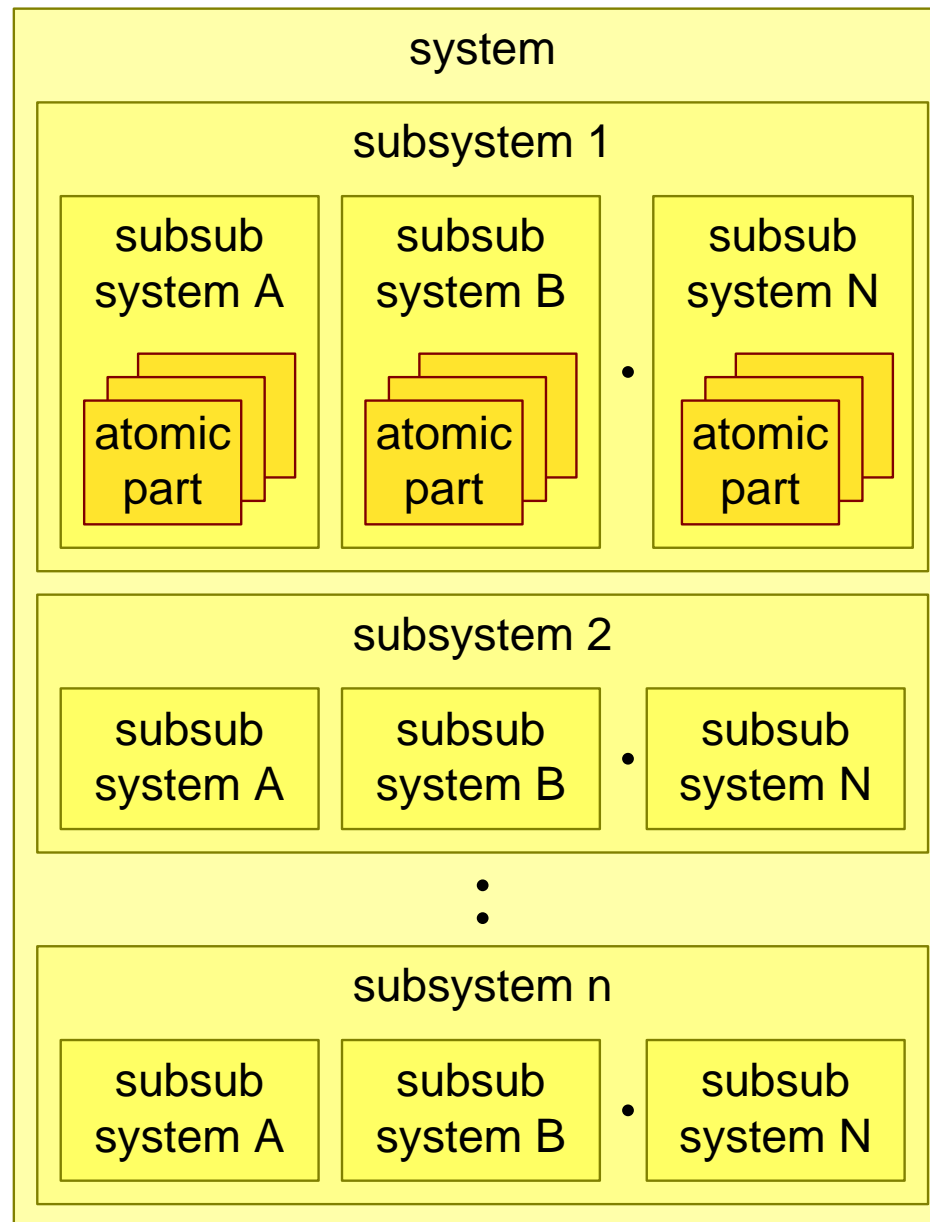
Engineering



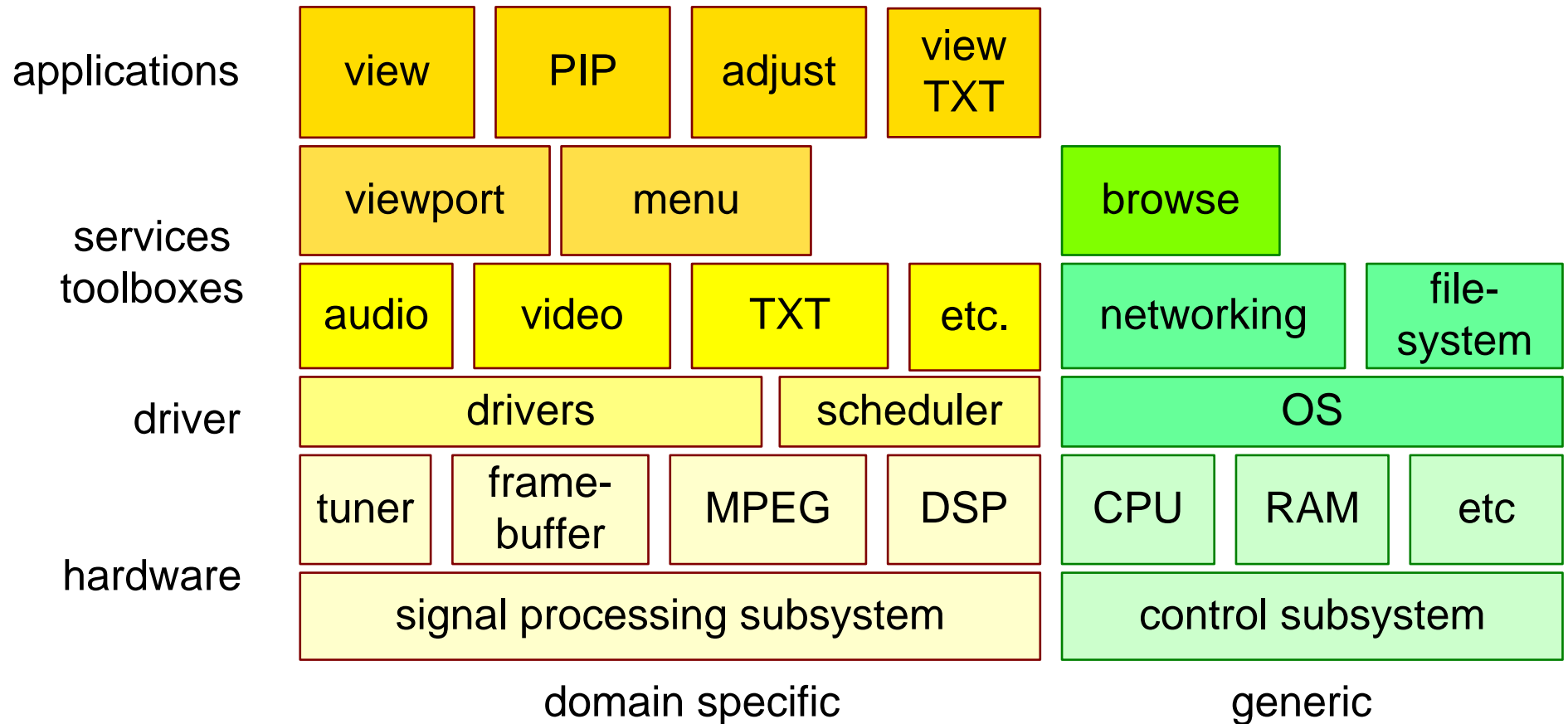
Example Physical Decomposition



Partitioning is Applied Recursively



Software plus Hardware Decomposition



Guidelines for Partitioning

the part is cohesive

functionality and technology belongs together

the coupling with other parts is minimal

minimize interfaces

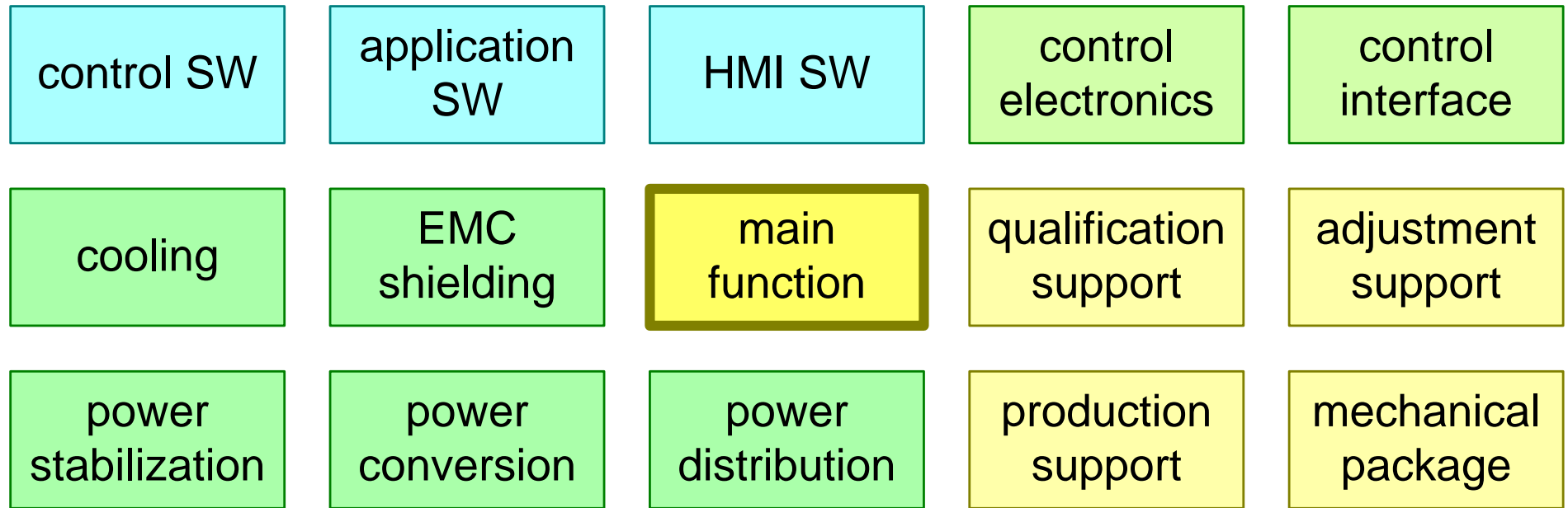
the part is selfsustained for production and qualification

can be in conflict with cost or space requirements

clear ownership of part

e.g. one department or supplier

How much self-sustained?



How self sustained should a part be?

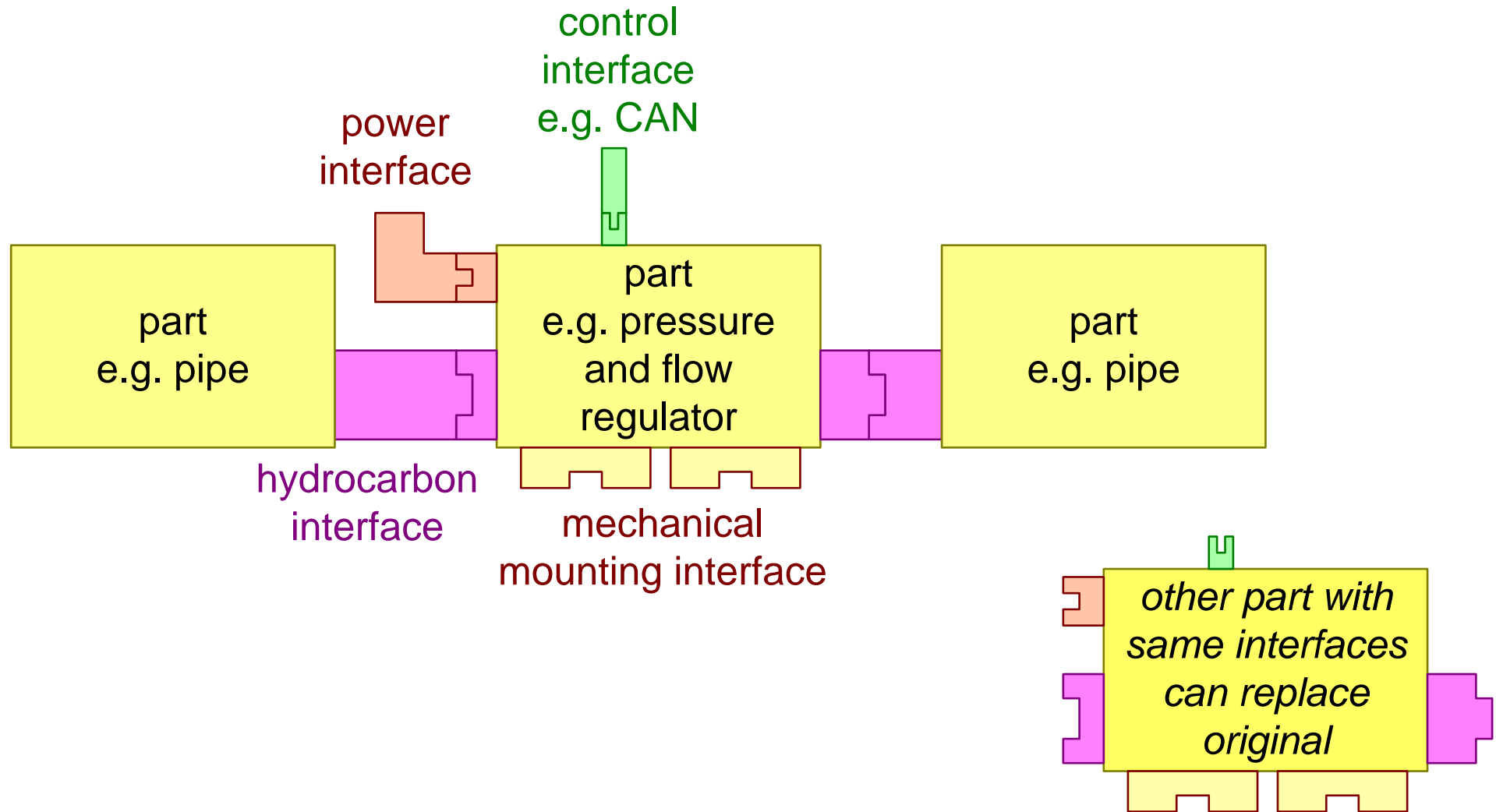
trade-off:

cost/speed/space
optimization



logistics/lifecycle/production
flexibility
clarity

Decoupling via Interfaces



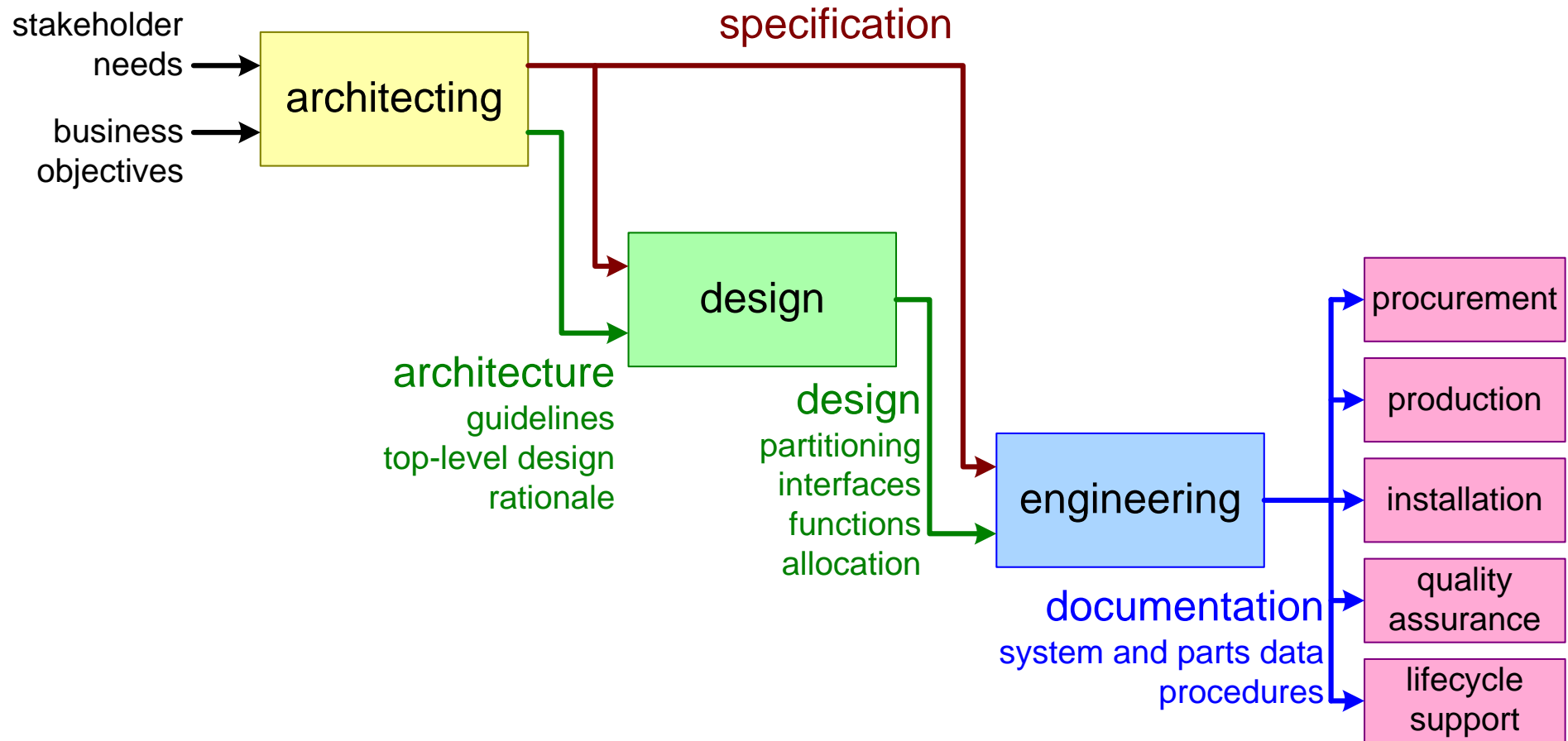
The Ideal Modularity

System is composed

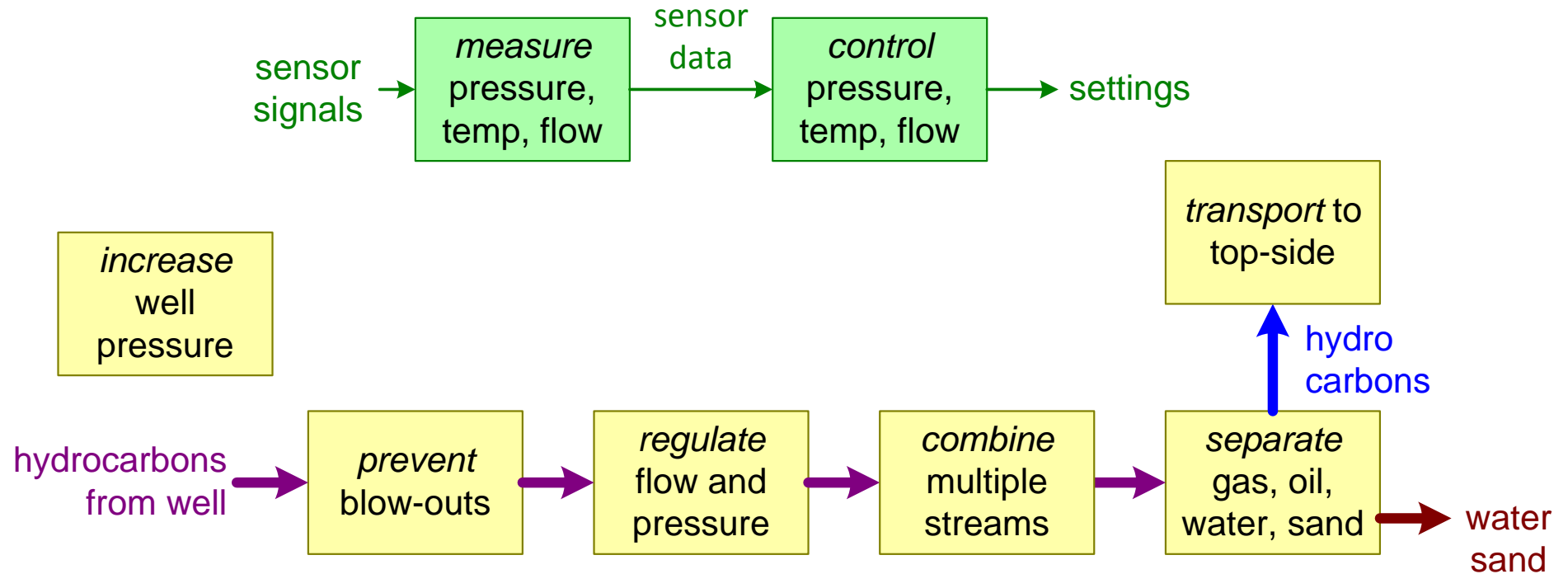
by using standard interfaces

limited catalogue of variants (e.g. cost performance points)

System Creation



Simplistic Functional SubSea Example



Functional Decomposition

How does the system work and operate?

Functions describe *what* rather than *how*.

Functions are *verbs*.

Input-Process-Output paradigm.

Multiple kinds of flows:

- physical (e.g. hydrocarbons)

- information (e.g. measurements)

- control

At lower level one part \approx one function

- pump pumps, compressor compresses, controller controls

At higher level functions are complex interplay of physical parts

- e.g. regulating constant flow, pressure and temperature

Quantification

Size 2.4m * 0.7m * 1.3m

Weight 1450 Kg

Cost 30000 NoK

Reliability MTBF 4000 hr

Throughput 3000 l/hr

Response time 0.1 s

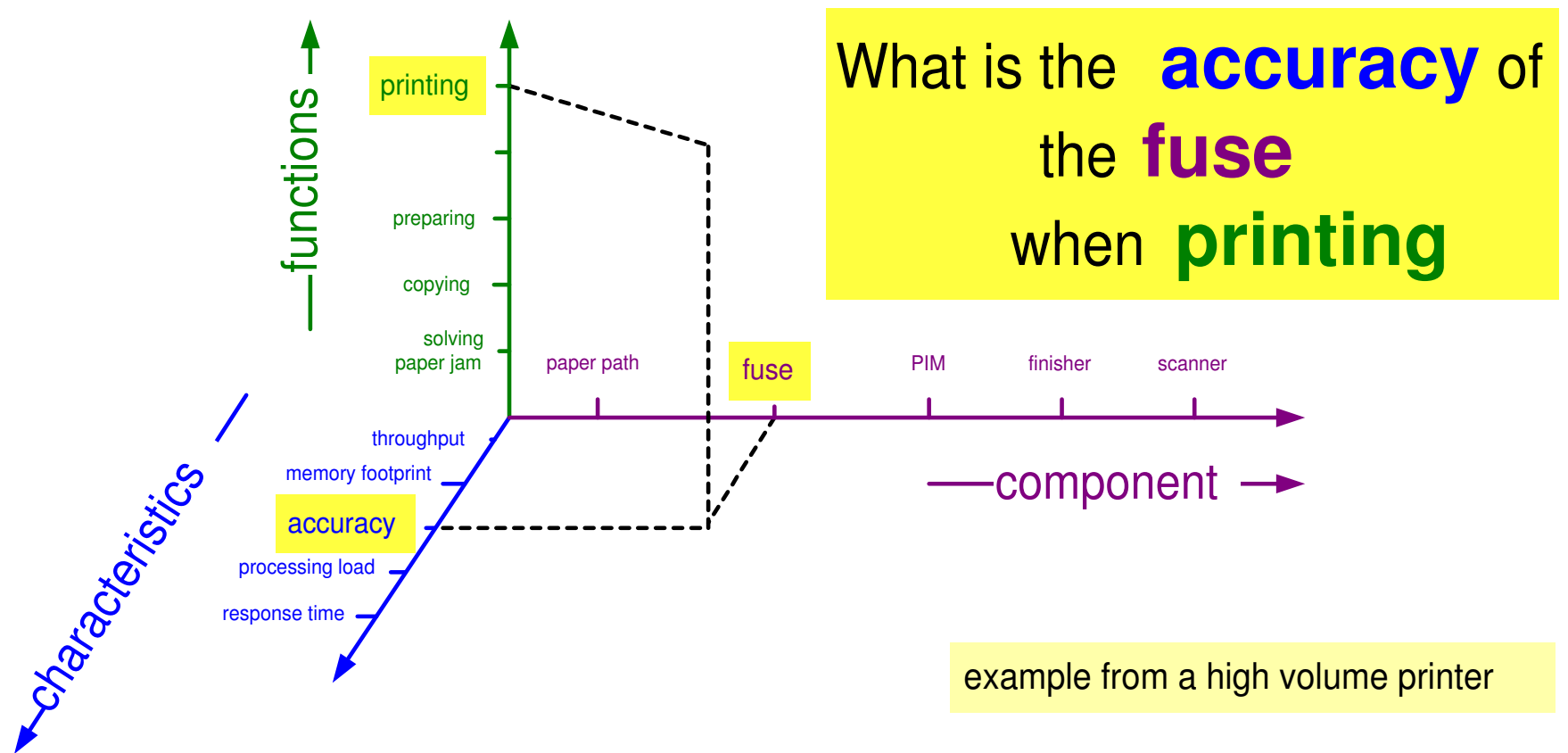
Accuracy +/- 0.1%

*many characteristics
of a system, function or part
can be quantified*

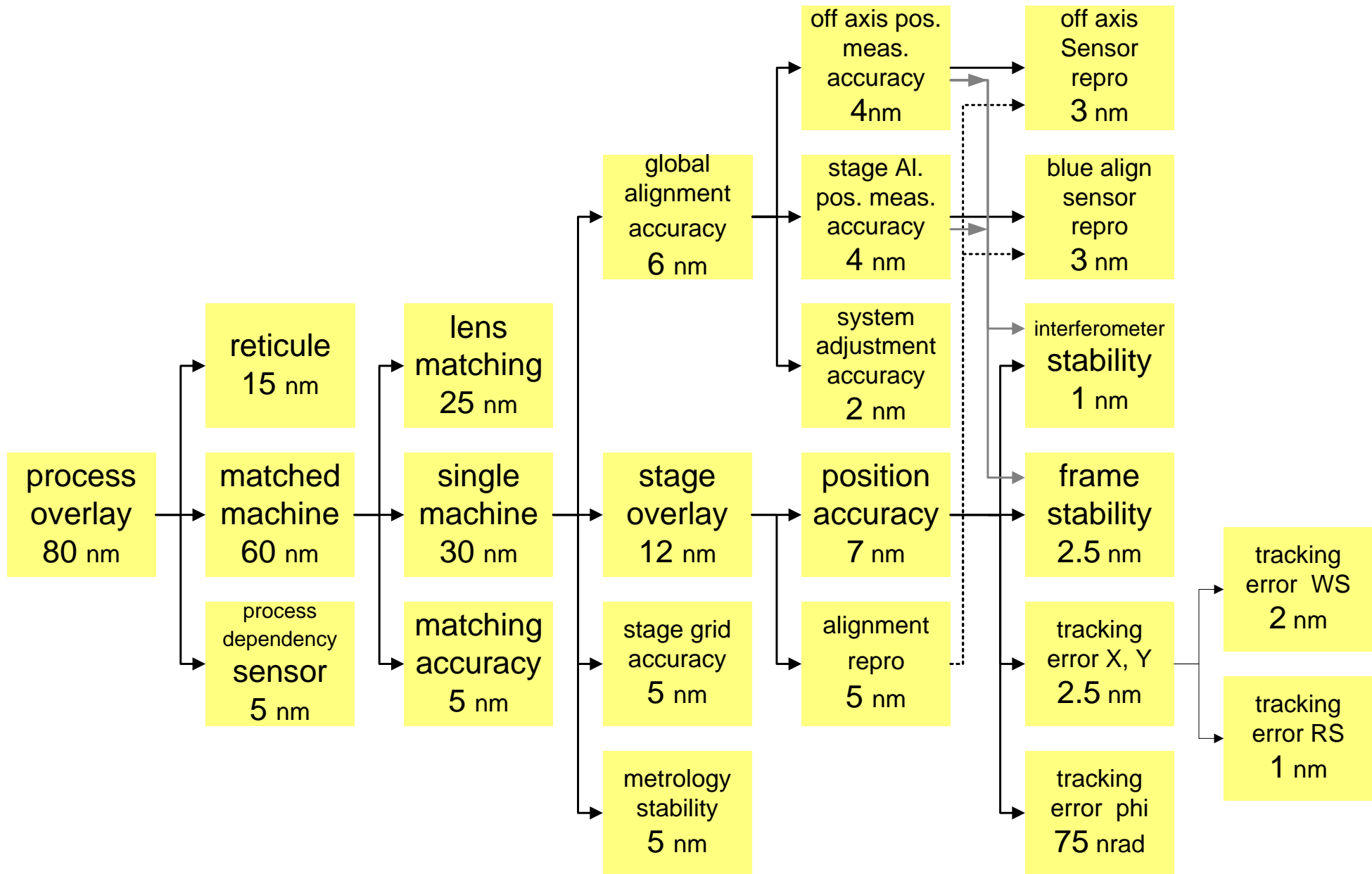
*Note that quantities
have **unit***

Question Generator

How about the **<characteristic>**
of the **<component>**
when performing **<function>**?

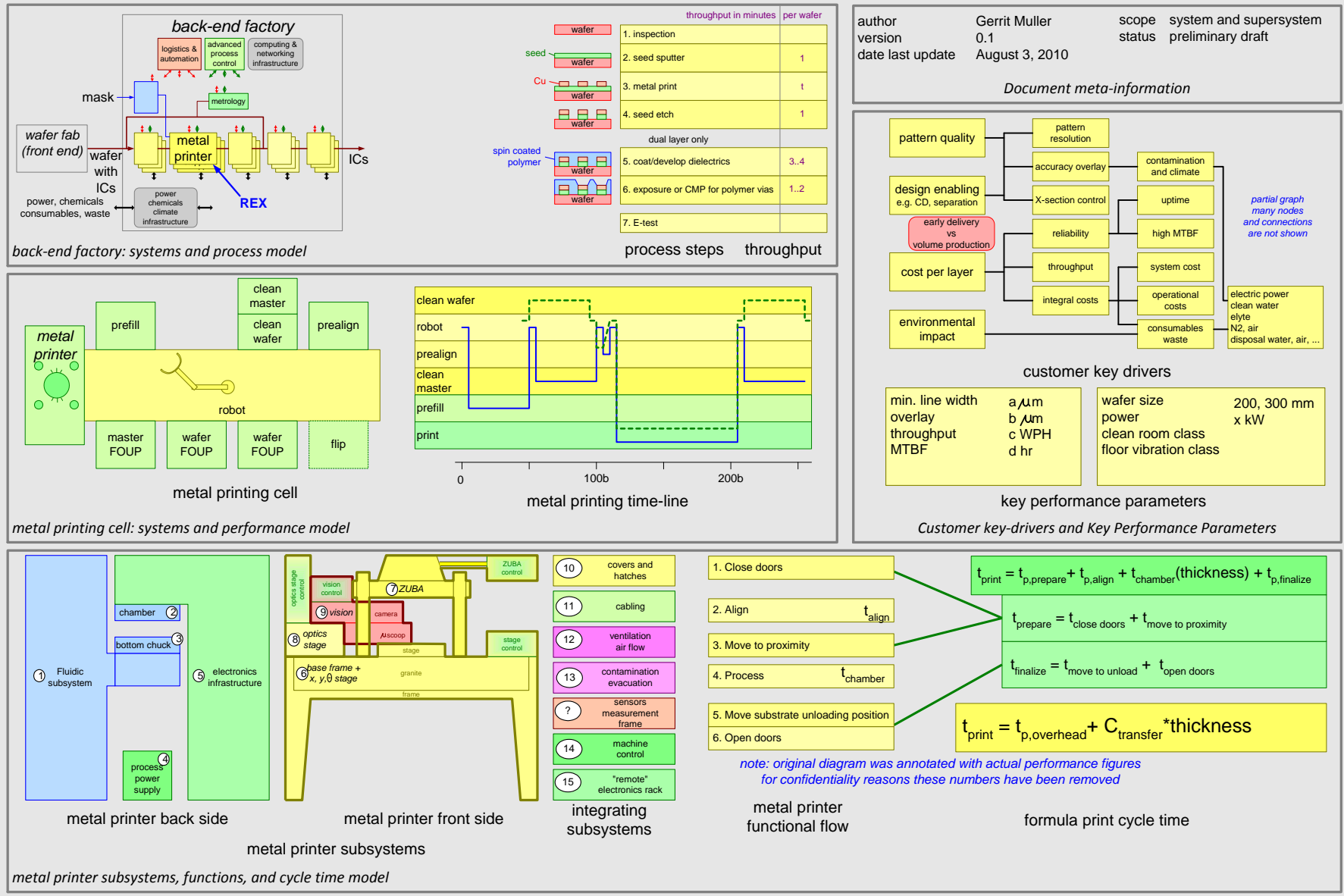


Example Technical Budget



Example of A3 overview

A3 architecture overview of the Metal Printer (all numbers have been removed for competitive sensitivity)



Exercise Dynamic Behavior

Capture the **dynamic behavior** of the **internals** of your system in **multiple** diagrams.

Diagrams that capture dynamic behavior are among others:

- Functional flow (of control or information, material or goods, or energy)
- Activity or sequence diagrams (e.g. with “swimming lanes”)
- State diagrams

Exercise Block Diagram

Make a set of **block diagrams** capturing the **static parts** and **interfaces**.

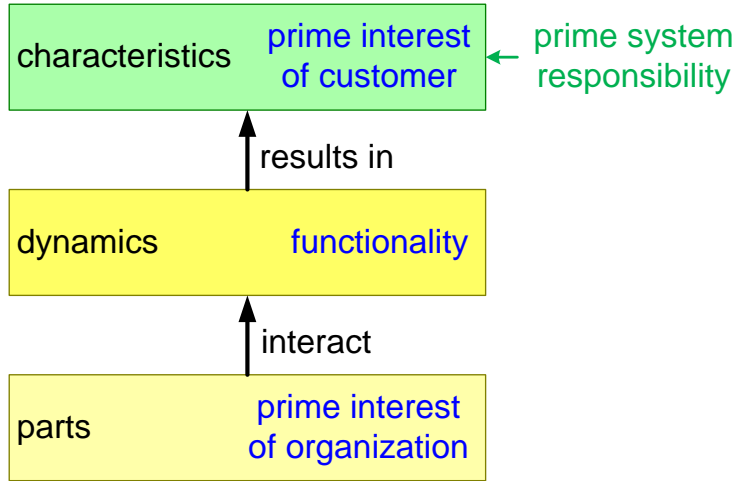
Ensure coverage of the entire system, e.g. including service, training, production, etc.

Show both **hardware** and **software**

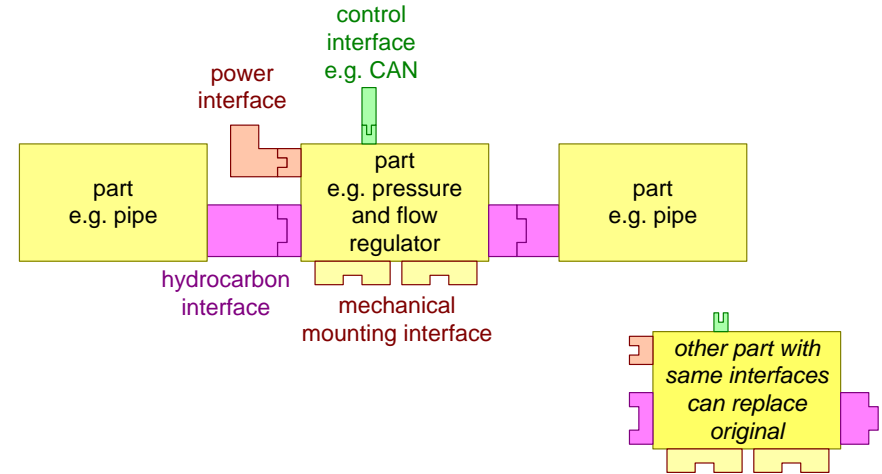
Good block diagrams have in the order of 10 to 20 blocks

Design Fundamentals

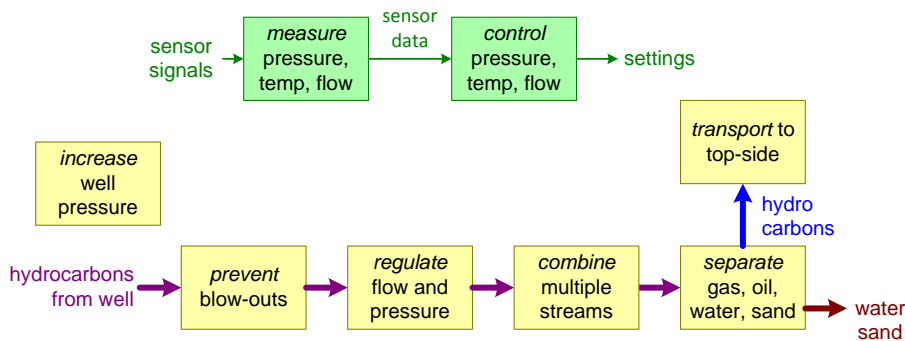
Parts, Dynamics, Characteristics



Decoupling via Interfaces



Dynamic Behavior



Question Generator

How about the **<characteristic>** of the **<component>** when performing **<function>**?

