TECHNISCHE UNIVERSITÄT DRESDEN

**Fakultät Informatik** Institut Software- und Multimediatechnik, Lehrstuhl Softwaretechnologie

# Life-by-Wire

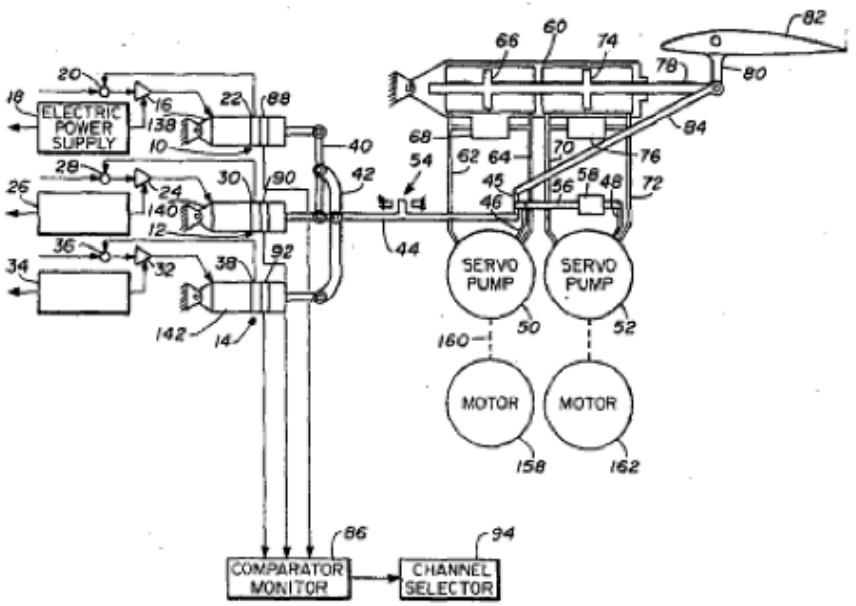
## with Cyber-Physical Systems

Prof. Dr. Uwe Aßmann
Technische Universität Dresden
Software Engineering Group
http://st.inf.tu-dresden.de
Linköping, May 2, 2012


ResUbic Lab
http://www.resubic.org

# Chap.1 The Presence: Fly-by-Wire

- **FLY-BY-WIRE**
- William G. Redmond - US Patent 3,679,156, 1972
- http://www.google.com/patents?hl=de&lr=&vid=USPAT3679156&id=GTswAAAAEBAJ&oi=fnd&dq=fly-by-wire&printsec=abstract#v=onepage&q=fly-by-wire&f=false



## Quality assurance

- Extensive testing

## Verification

- Normal mode
- Rescue mode

## Certification

- Certification agencies
- For product liability

# Chap.2 The Near Future: Help-By-Wire

http://de.wikipedia.org/wiki/Ungl%C3%BCck_bei_der_Loveparade_2010

- „In der Europäischen Union werden jährlich ueber 7.000 Fussgänger und 2.000 Fahrradfahrer bei Verkehrsunfällen getätet. Hunderttausende werden darüber hinaus bei Verkehrsunfällen verletzt."
- [M. Bischoff. Aktive Sicherheitssysteme fuer den Schutz von Fussgaengern im Strassenverkehr]

Um 4.30 Uhr hatte sich Regina F. auf den Weg zur Arbeit gemacht. Von der Britzer Straße in Schöneweide zum Putzen nach Marienfelde. Sie schaffte es nicht mal bis zum S-Bahnhof.

Viel zu schnell kam ein Kombi um die Ecke gefahren. So schnell, dass das Auto aus der Kurve flog und über den Bürgersteig raste. Durch den Aufprall wurde Regina F. auf die Straße geschleudert. Nur kurz hielt der Mann an, dann gab er Gas und raste davon.

http://www.bz-berlin.de/bezirk/treptow/raser-faehrt-oma-um-und-macht-sich-davon-article1373880.html

**Passive Fail-safe system (FS)**

- reach a safe state without power

**Quality assurance**

- Extensive testing

**Verification**

- Normal mode
- Safe

**Certification**

- Certification agencies
- For product liability

# Chap.3 The Mid-Term Future: Move-By-Wire

http://commons.wikimedia.org/wiki/File:Traffic_seen_from_top_of_Arc_de_Triomphe.JPG


https://upload.wikimedia.org/wikipedia/commons/thumb/1/14/Google_self_driving_car_at_the_Googleplex.jpg/800px-Google_self_driving_car_at_the_Googleplex.jpg

Prof. U. Aßmann, TU Dresden

[Acatech]

# Requirements for Move-by-Wire Systems

**Active Fail-safe systems (AFS)**

- reach this safe state only with power

**Fail-operational systems (FO)**

- may fail but continue to work

**Quality assurance**

- Extensive testing

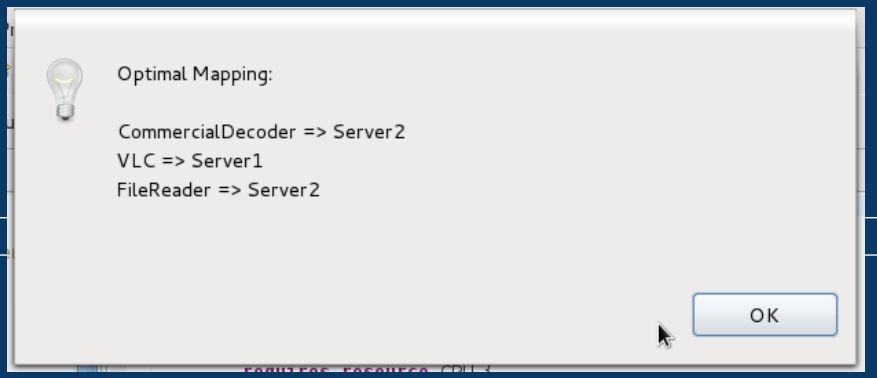
**Verification**

- Normal mode
- Safe

**Certification**

- Certification agencies
- For product liability

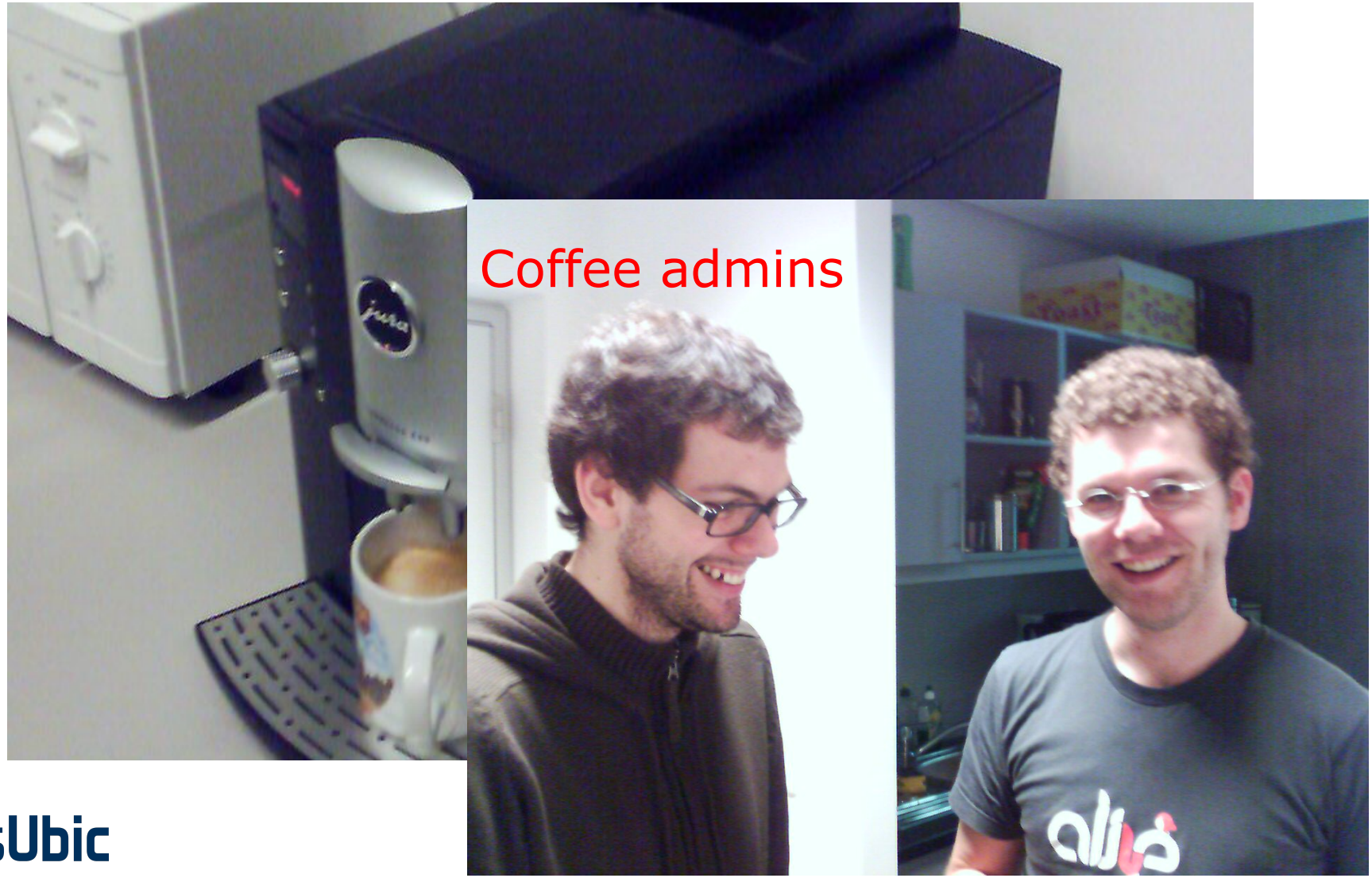
TECHNISCHE UNIVERSITÄT DRESDEN




# Chap.4. The Software for Life-by-Wire: Multi-Objective Optimization Architectures (MOO Architectures)

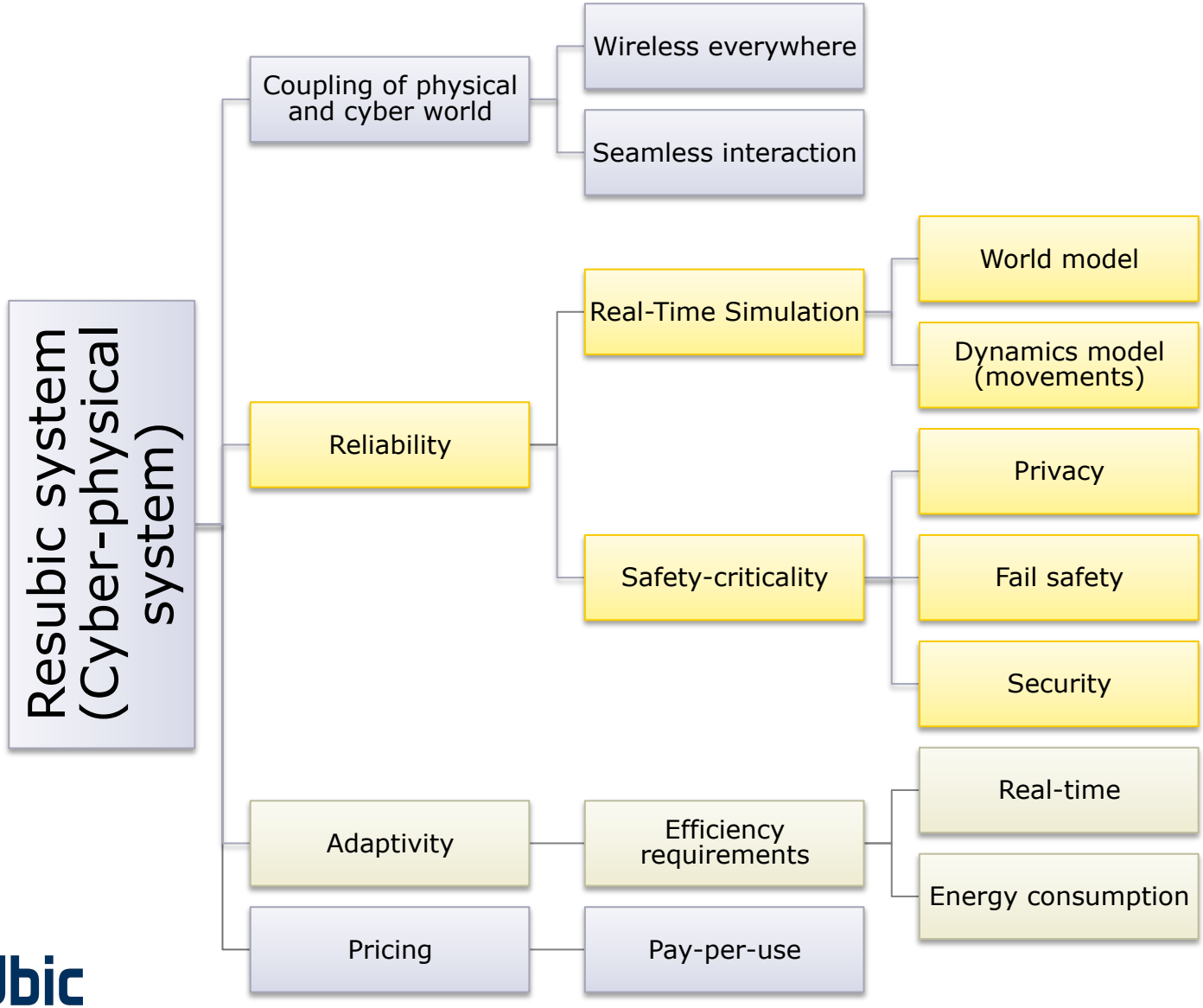
Prof. Uwe Aßmann
Sebastian Götz

# The eternal problems...

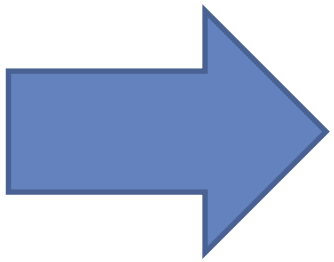


# Recently, we got a NAO Robot „Chuck“

# Requirements for Life-by-Wire

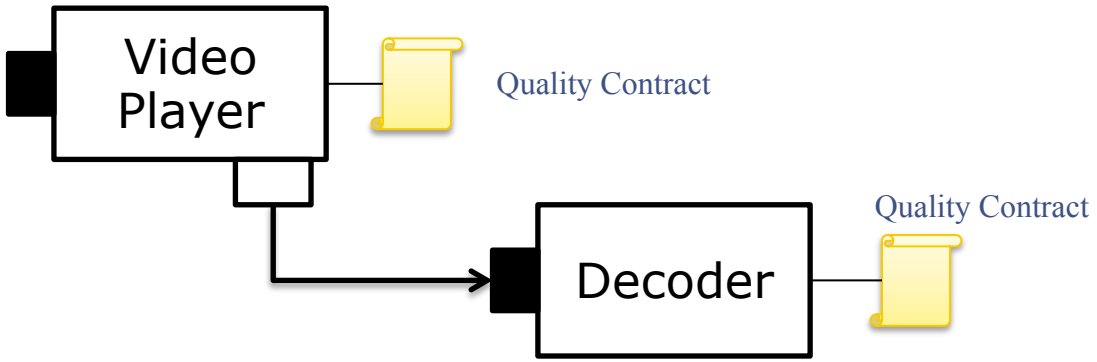


- CPS need adaptability
  - QoS adaptation
  - Context adaptation
- CPS must be reliably composable
  - Functional Contracts
  - Quality Contracts
    - Real-time contracts
    - Energy contracts
    - Safety contracts
    - Spatial dynamics contracts
  - Energy reserve for active fail-safety (AFS)



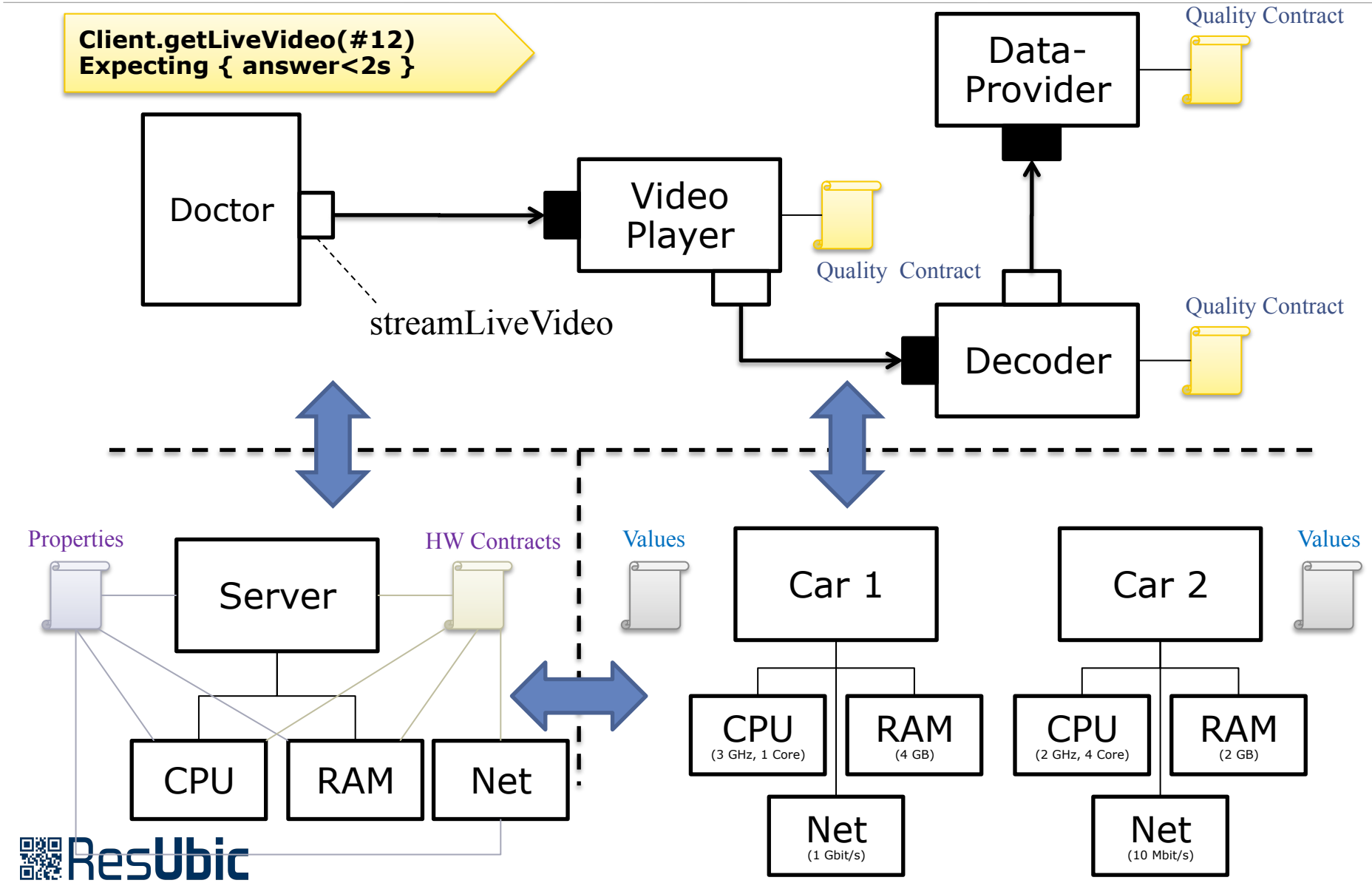
- Contract Checking of Quality Properties
- Configuration Optimization
  - Determining Optimal Mappings of Software Components onto Hardware Resources
  - Resource minimization (energy)
  - Computation of pricing

- **Temporal path-based quality contracts** are constracts by which the test coverage can be decided statically
- Quality contracts allow for writing statements over time that are verified
  - „in 5ms the system is in a safe state"
  - „the system is life and will never get stuck"
  - „After the door is closed, the train will eventially run"
  - „The robot swarm can still run 25s, so the shutdown to safe state has start in 5s"
  - „This robot cannot be functionally replaced – function of the swarm fails
- Quality contracts assure that quality features remain during composition of components

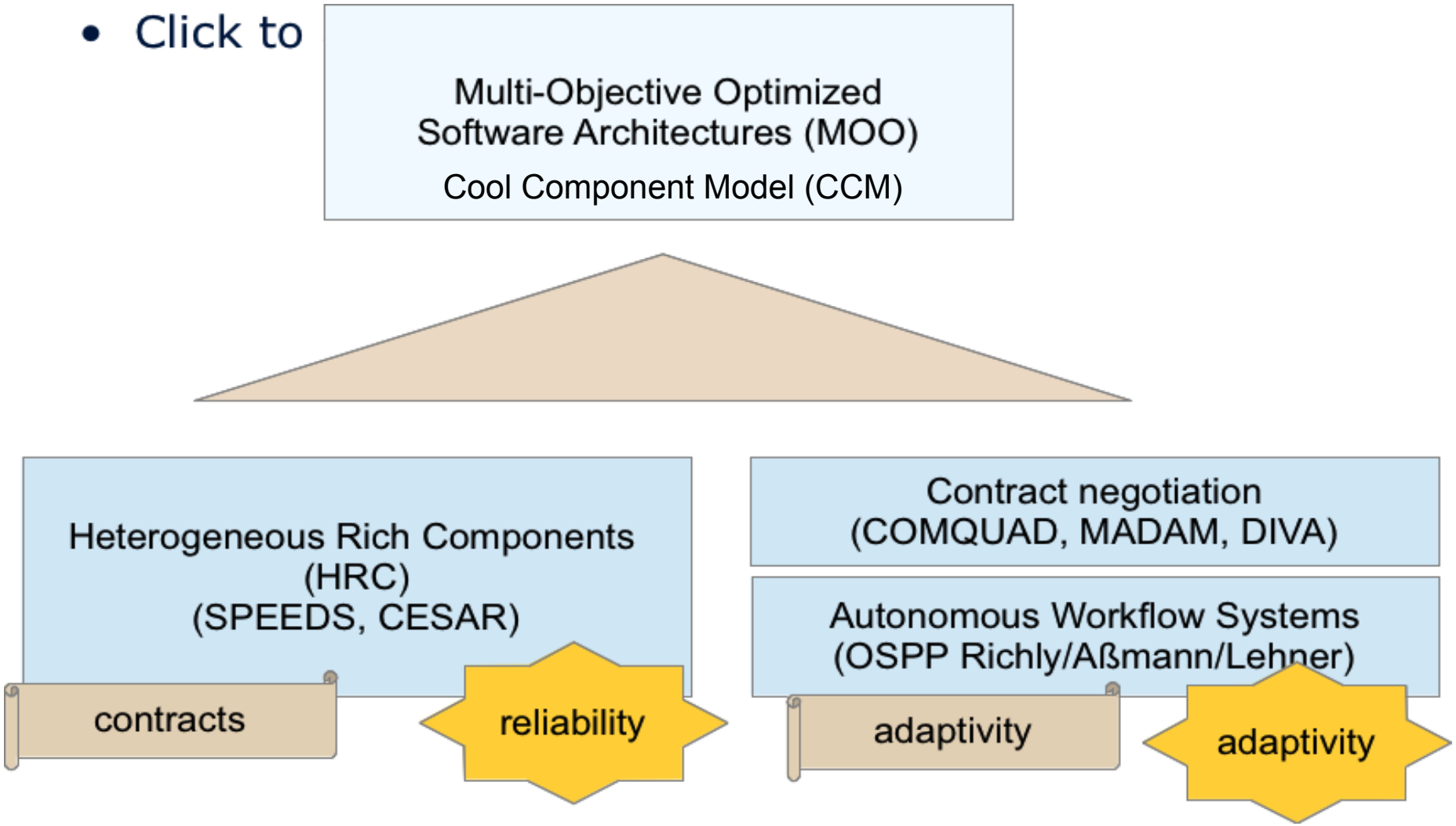


# Quality Contracts in a Car Rescue Application

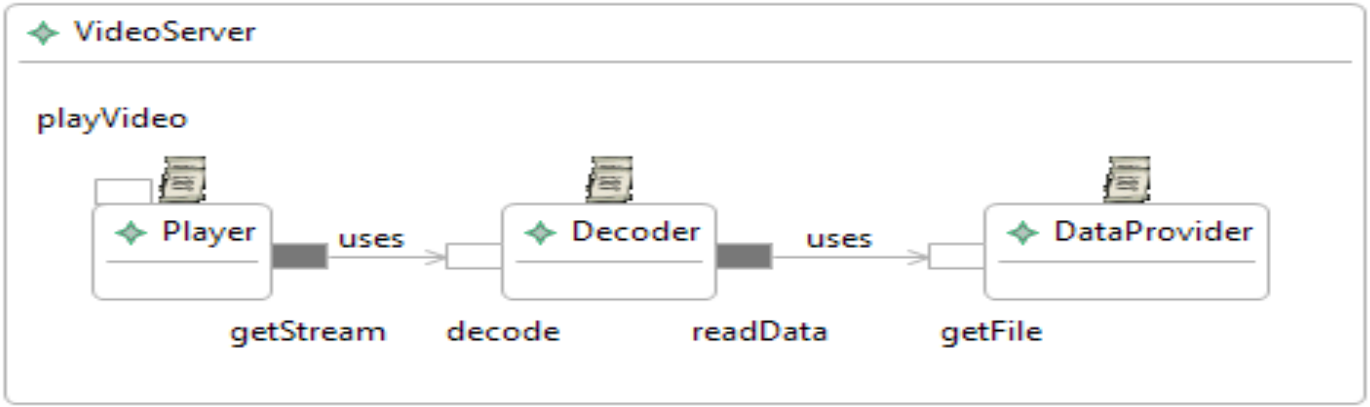




- Click to


Multi-Objective Optimized
Software Architectures (MOO)
Cool Component Model (CCM)
Heterogeneous Rich Components
(HRC)
(SPEEDS, CESAR)
contracts
reliability
Contract negotiation
(COMQUAD, MADAM, DIVA)
Autonomous Workflow Systems
(OSPP Richly/Aßmann/Lehner)
adaptivity
adaptivity

- Software is described using a CCM Structural Model
  - approx. SysML plus QCL contract language




- SW Components have provided/required ports (e.g. methods of classes)
- Quality-Modes and Implementations of SW Components are described using QCL contracts

VideoServer_Player.ecl

```
1  import ccm [../VideoServer.structure]
2  import ccm [../Server.structure]
3
4  contract VLC implements software Player {
5      mode highQuality {
6          requires component Decoder {
7              bitrate min: 5
8          }
9          requires resource CPU {frequency min: 1500}
10
11         provides framerate min: 20
12     }
13     mode lowQuality {
14         requires component Decoder {
15             bitrate min: 2
16         }
17         requires resource CPU {frequency
18
19         provides framerate min: 10
20     }
21 }
```

**Quality Modes**

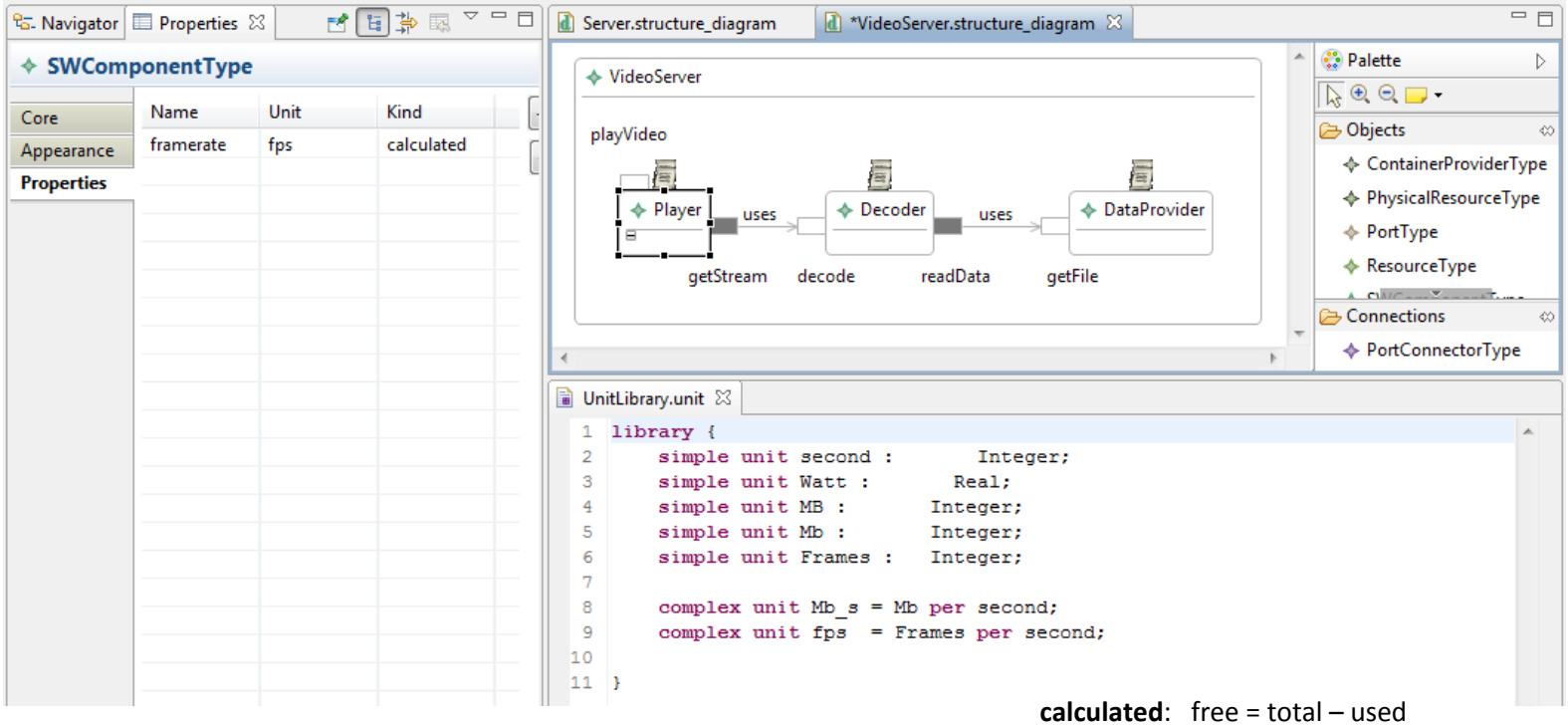
- SW Dependencies
- HW Dependencies
- Provided Properties

*Where do the properties come from?*

*Where and how to define HW ?*

VideoServer_Player.ecl | VideoServer_DataProvider.ecl

```
1  import ccm [../VideoServer.structure]
2  import ccm [../Server.structure]
3
4  contract FileReader implements software DataProvider {
5      mode file {
6          provides throughput min: 20
7          requires resource HDD {
8              throughput min: 20
9          }
10     }
11 }
12
13 contract URLReader implements software DataProvider {
14     mode url {
15         provides throughput min: 5
16         requires resource Network {
17             bandwidth min: 5
18         }
19     }
20 }
```

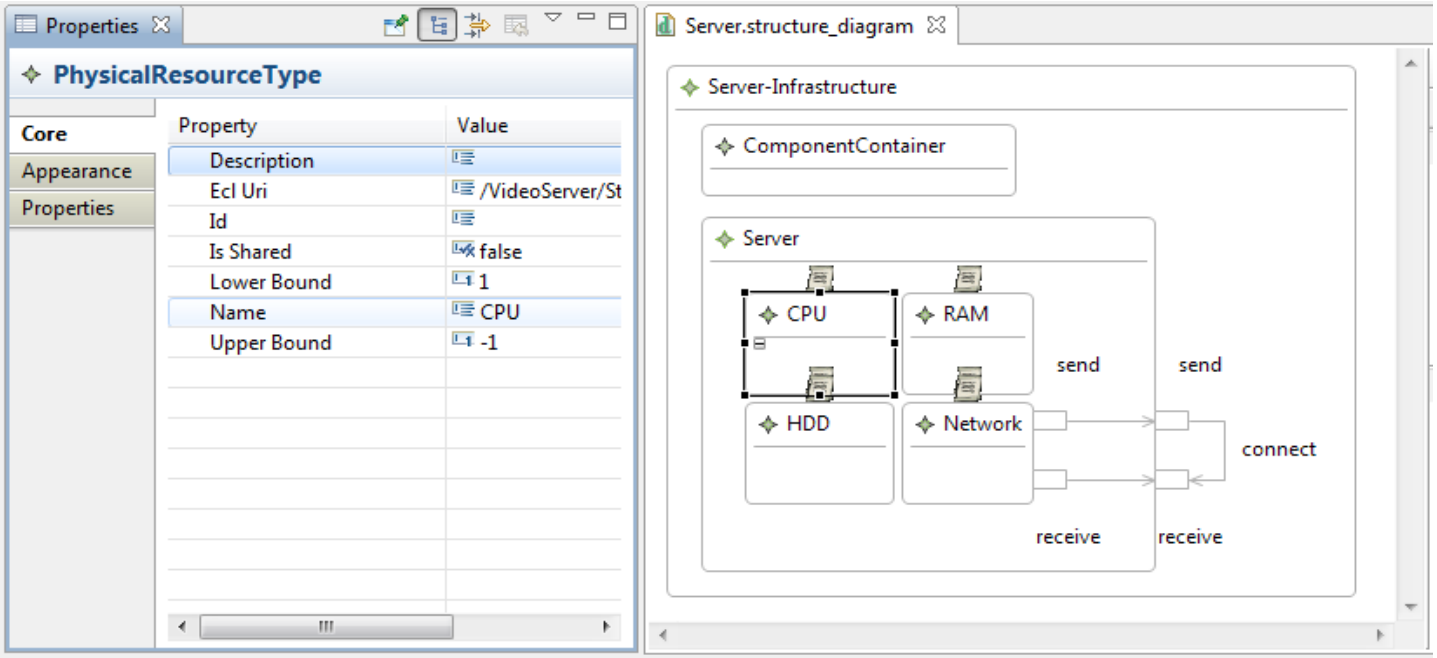



**calculated**: free = total – used
**monitored**: ResourceMgr.getX()
**static**: fixed value for instance (e.g., size of HDD)

- Properties are defined for each ComponentType.
- Units can be defined
- They are either calculated, monitored or static.

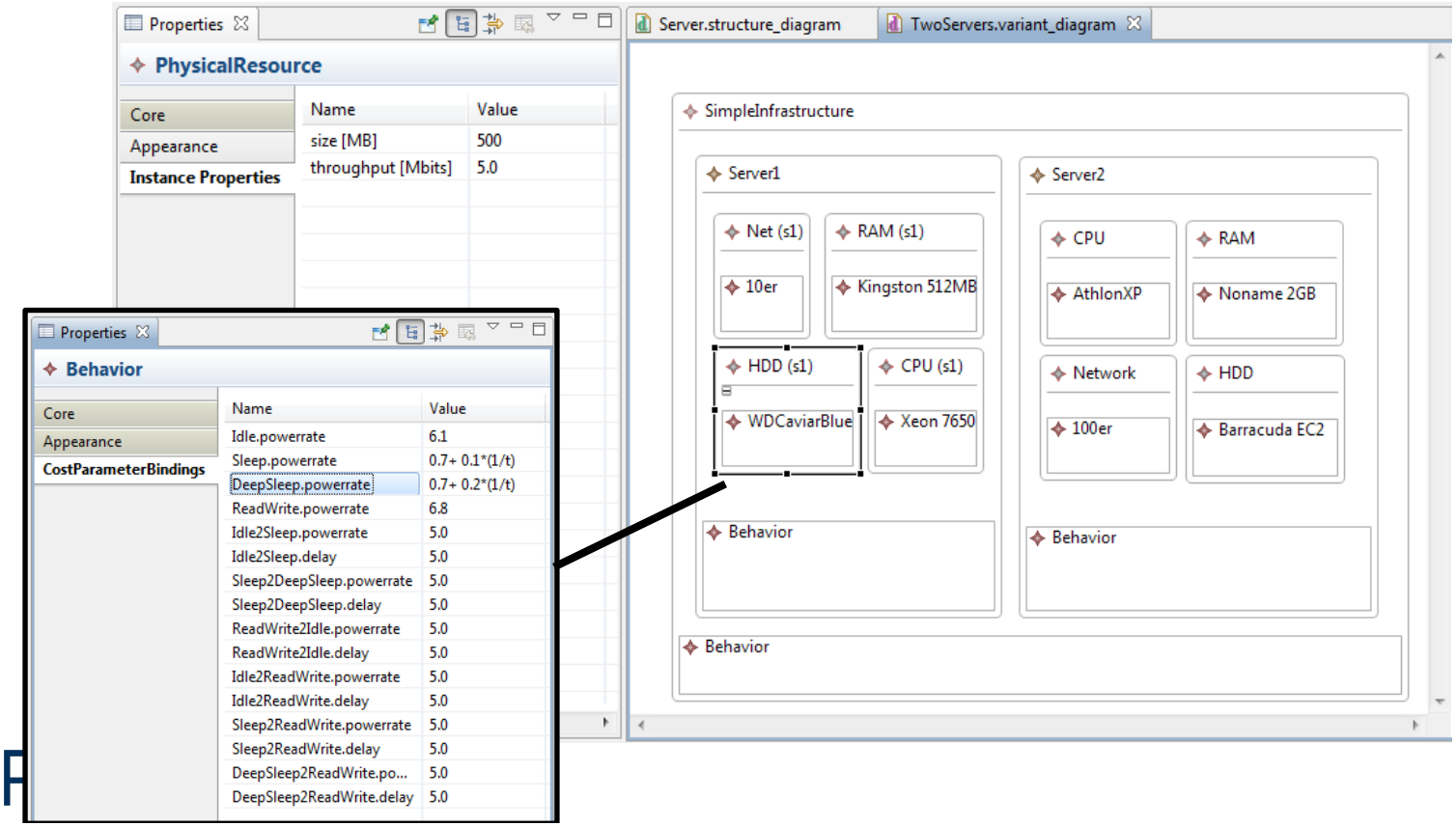
- Which ResourceTypes exist
- How they are connected
- Multiplicities - Contracts

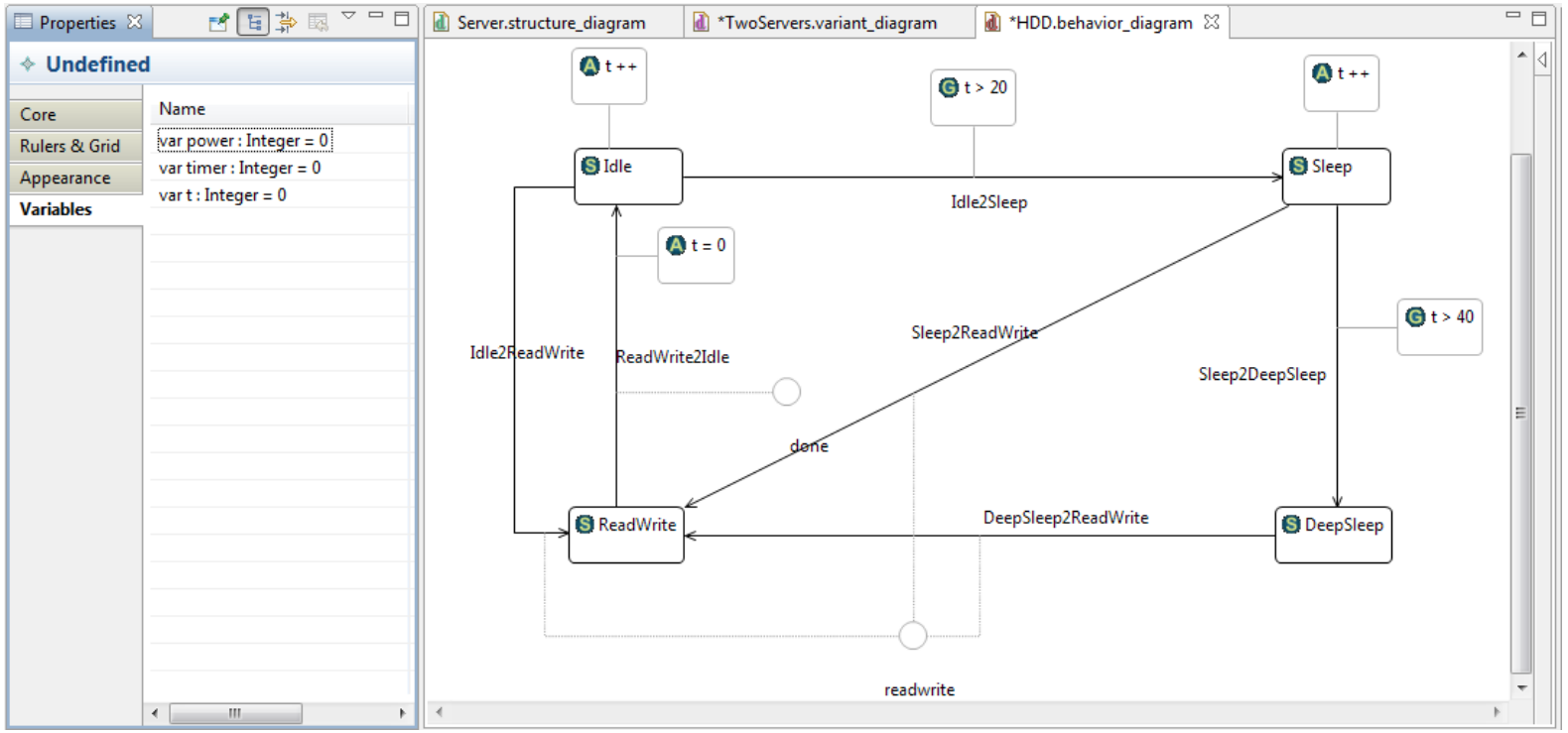
- Hardware Structural Model




- Hardware Variant Model Concrete Resources (next Slide)
- Hardware Behavior using Energy State Charts

- concrete resources are modeled and connected to structural model
- values of static properties are assigned per resource
- behavior templates are assigned per resource,
- Cost parameters are assigned with values or formulas






- use cost parameters instead of concrete values
  - reuse for resources of same kind
- can be executed using workload descriptions

ResUbic

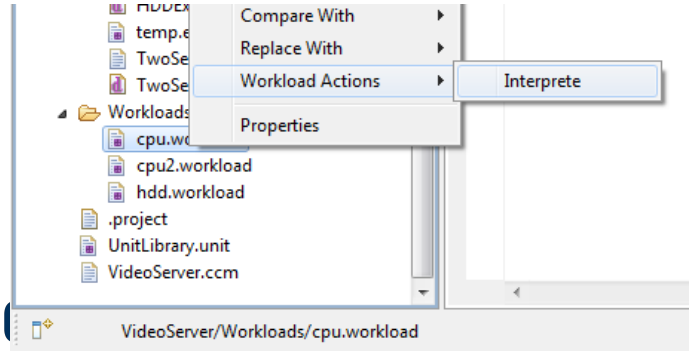
# Workload Descriptions

*HDD.behavior_dia... | hdd.workload | HDDExampleTest.va...

```
1  import ccm [./VideoServer/Variants/HDDExampleTest.variant]
2
3  Workload TestHDD for WDHDD {
4      readwrite at 2
5      done at 5
6      readwrite at 20
7      done at 30
8      readwrite at 60
9      done at 100
10     readwrite at 150
11     done at 155
12 }
```

hdd.workload | cpu2.workload | cpu.workload

```
1  import ccm [./VideoServer/Variants/CPUExampleTest.variant]
2
3  Workload someName for Xeon {
4      work at 5
5      work at 20 for 15
6      work at 50 for 20
7  }
```

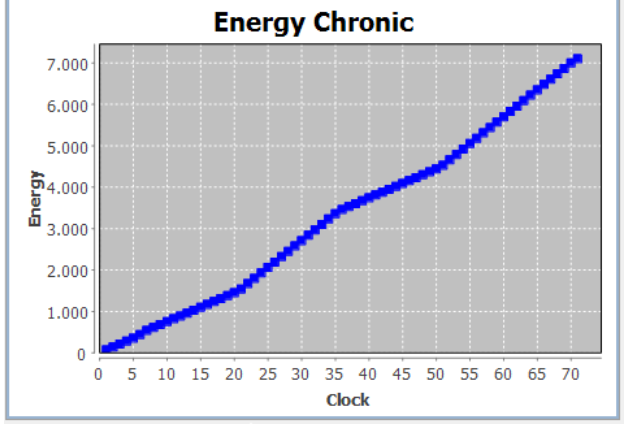


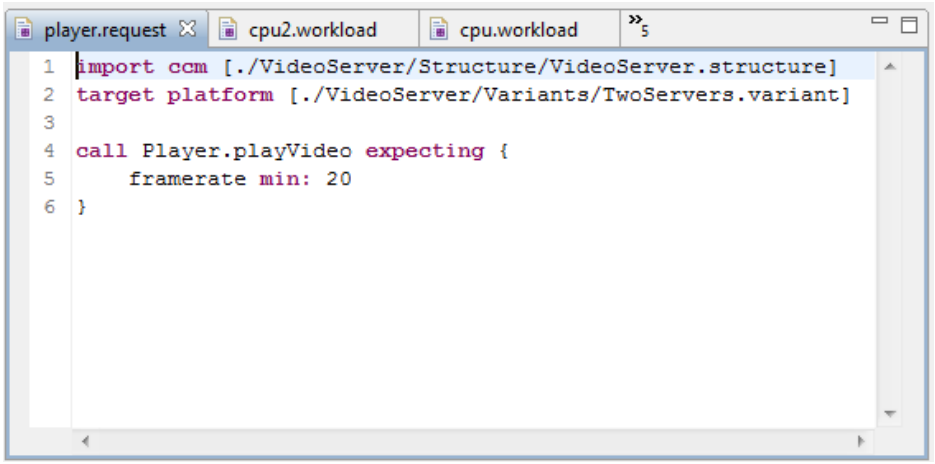

Simulation Results

CPU

| Clock | Power | State | Config |
|---|---|---|---|
| 1 | 70 | Idle | t = 1, x = |
| 2 | 140 | Idle | t = 2, x = |
| 3 | 210 | Idle | t = 3, x = |
| 4 | 280 | Idle | t = 4, x = |
| 5 | 350 | Idle | t = 5, x = |
| 6 | 440 | Busy | t = 5, x = 1 |
| 7 | 540 | Idle | t = 1, x = 1 |
| 8 | 610 | Idle | t = 2, x = 1 |
| 9 | 680 | Idle | t = 3, x = 1 |
| 10 | 750 | Idle | t = 4, x = 1 |
| 11 | 820 | Idle | t = 5, x = 1 |
| 12 | 890 | Idle | t = 6, x = 1 |
| 13 | 960 | Idle | t = 7, x = 1 |
| 14 | 1030 | Idle | t = 8, x = 1 |
| 15 | 1100 | Idle | t = 9, x = 1 |
| 16 | 1170 | Idle | t = 10, x = 1 |
| 17 | 1240 | Idle | t = 11, x = 1 |
| 18 | 1310 | Idle | t = 12, x = 1 |
| 19 | 1380 | Idle | t = 13, x = 1 |
| 20 | 1450 | Idle | t = 14, x = 1 |
| 21 | 1540 | Busy | t = 14, x = 2 |
| 22 | 1670 | Busy | t = 14, x = 3 |
| 23 | 1800 | Busy | t = 14, x = 4 |
| 24 | 1930 | Busy | t = 14, x = 5 |
| 25 | 2060 | Busy | t = 14, x = 6 |
| 26 | 2190 | Busy | t = 14, x = 7 |
| 27 | 2320 | Busy | t = 14, x = 8 |
| 28 | 2450 | Busy | t = 14, x = 9 |
| 29 | 2580 | Busy | t = 14, x = 10 |
| 30 | 2710 | Busy | t = 14, x = 11 |
| 31 | 2840 | Busy | t = 14, x = 12 |

Energy Chronic | Problems






- Users call methods on SW Components
- They state their (quality) expectations explicitly
- HW infrastructure is selected

- Then the software configuration is automatically optimized with Multi-Objective-Optimization (MOO)

# Multi-Objective Optimization (MOO) of Configurations

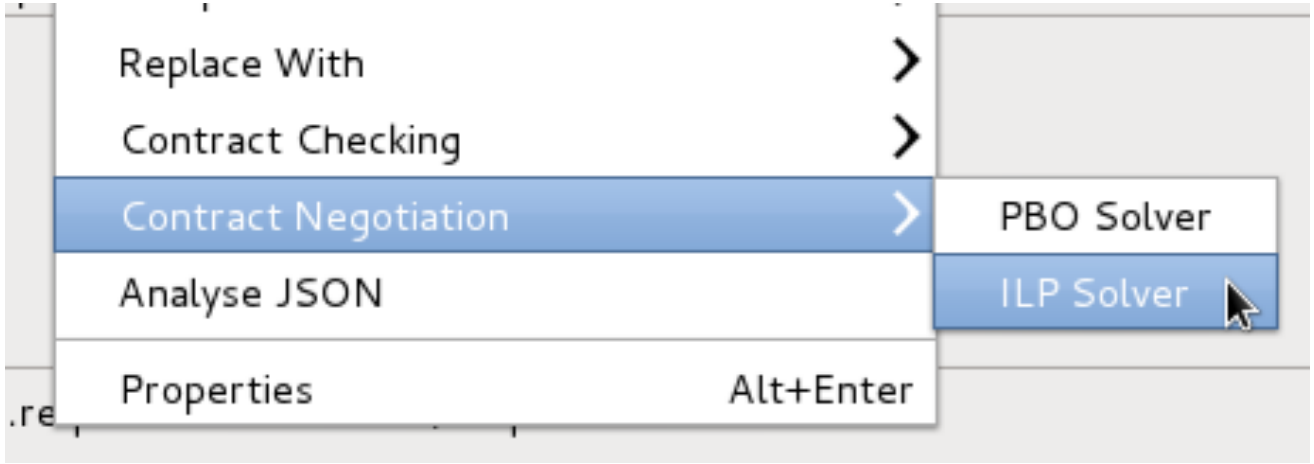


Question:

- Which implementations of the contracts
- in which quality mode
- on which resources
- give the most efficient configuration?

Multi-Objective Optimization (MOO) with:

- Integer Linear Programming (exact)
- Pseudo-Boolean Optimization (exact)
- Ant Colony Optimization (approx.)
- Simulated Annealing (approx.)
- ...

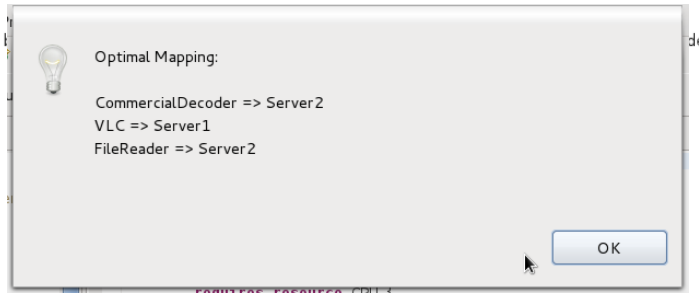



- An ILP is generated for a certain request on a software component and a hardware variant.

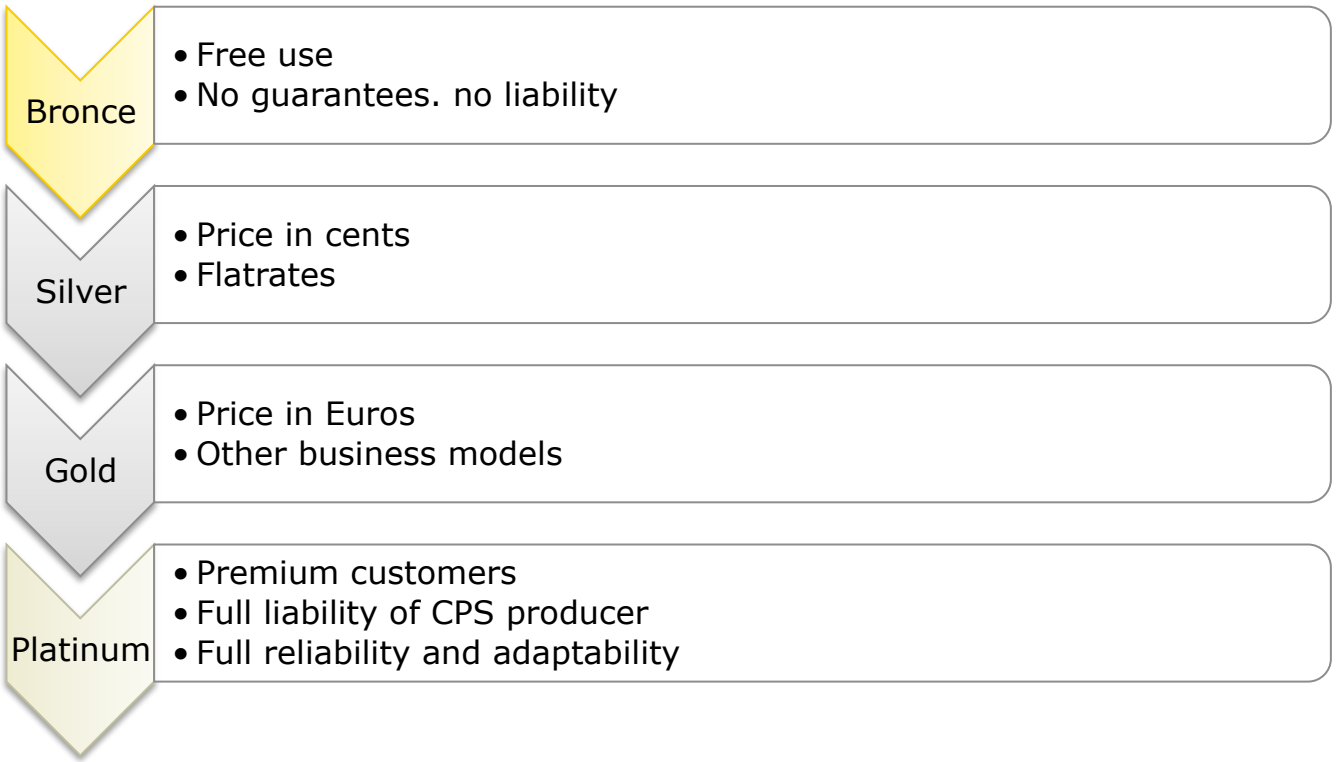
# The Generated ILP

VideoServer_Decoder.ecl | player.request.lp

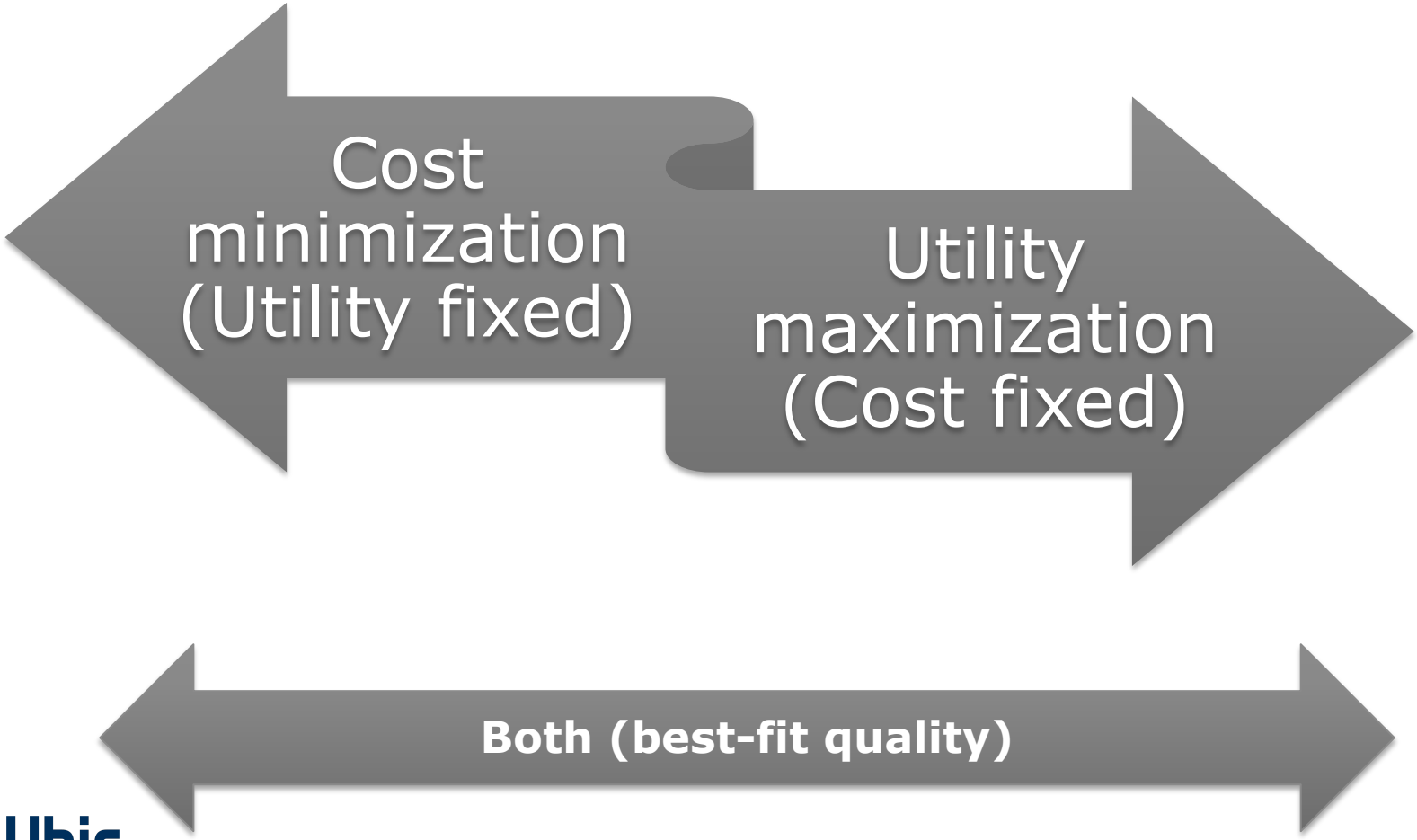
```
min: usage#Server1#CPU_[s1]#frequency + usage#Server2#CPU#frequency + usage#Server1#Net_[s1]#bandwidth + usage#Server1#HDD_[s1]#size + usage#Server2#Network#bar

b#VLC#highQuality#Server2 + b#VLC#highQuality#Server1 + b#VLC#lowQuality#Server2 + b#VLC#lowQuality#Server1 = 1;
b#FreeDecoder#fast#Server2 + b#FreeDecoder#fast#Server1 + b#FreeDecoder#slow#Server2 + b#FreeDecoder#slow#Server1 + b#CommercialDecoder#ultrafast#Server2 + b#Co
b#FileReader#file#Server2 + b#FileReader#file#Server1 + b#URLReader#url#Server2 + b#URLReader#url#Server1 = 1;

usage#Server1#CPU_[s1]#frequency <= 2000.0;
usage#Server1#CPU_[s1]#frequency >= 0;
usage#Server1#CPU_[s1]#frequency = 1500.0 b#VLC#highQuality#Server1 + 1500.0 b#FreeDecoder#fast#Server1 + 1000.0 b#FreeDecoder#slow#Server1 + 800.0 b#Commercial
usage#Server1#RAM_[s1]#size <= 512.0;
usage#Server1#RAM_[s1]#size >= 0;
usage#Server1#RAM_[s1]#size = 512.0 b#FreeDecoder#fast#Server1 + 256.0 b#FreeDecoder#slow#Server1 + 128.0 b#CommercialDecoder#slow#Server1 + 512.0 b#CommercialD
usage#Server1#Net_[s1]#bandwidth <= 10.0;
usage#Server1#Net_[s1]#bandwidth >= 0;
usage#Server1#Net_[s1]#bandwidth = 5.0 b#URLReader#url#Server1;
usage#Server1#HDD_[s1]#size <= 500.0;
usage#Server1#HDD_[s1]#size >= 0;
usage#Server1#HDD_[s1]#throughput_w <= 5.0;
usage#Server1#HDD_[s1]#throughput_w >= 0;
usage#Server2#CPU#frequency <= 1000.0;
usage#Server2#CPU#frequency >= 0;
usage#Server2#CPU#frequency = 1500.0 b#VLC#highQuality#Server2 + 1500.0 b#FreeDecoder#fast#Server2 + 1000.0 b#FreeDecoder#slow#Server2 + 800.0 b#CommercialDeco
usage#Server2#RAM#size <= 2048.0;
usage#Server2#RAM#size >= 0;
usage#Server2#RAM#size = 512.0 b#FreeDecoder#fast#Server2 + 256.0 b#FreeDecoder#slow#Server2 + 128.0 b#CommercialDecoder#slow#Server2 + 512.0 b#CommercialDecode
usage#Server2#Network#bandwidth <= 100.0;
usage#Server2#Network#bandwidth >= 0;
usage#Server2#Network#bandwidth = 5.0 b#URLReader#url#Server2;
usage#Server2#HDD#size <= 150.0;
usage#Server2#HDD#size >= 0;
usage#Server2#HDD#throughput_w <= 25.0;
usage#Server2#HDD#throughput_w >= 0;
bitrate = 10.0 b#CommercialDecoder#fast#Server1 + 8.0 b#FreeDecoder#fast#Server2 + 3.0 b#FreeDecoder#slow#Server2 + 10.0 b#CommercialDecoder#fast#Server2 + 3.0
throughput = 5.0 b#URLReader#url#Server1 + 20.0 b#FileReader#file#Server2 + 5.0 b#URLReader#url#Server2 + 20.0 b#FileReader#file#Server1;
framerate = 10.0 b#VLC#lowQuality#Server1 + 10.0 b#VLC#lowQuality#Server2 + 20.0 b#VLC#highQuality#Server2 + 20.0 b#VLC#highQuality#Server1;
bitrate >= 2.0 b#VLC#lowQuality#Server1 + 2.0 b#VLC#lowQuality#Server2 + 5.0 b#VLC#highQuality#Server2 + 5.0 b#VLC#highQuality#Server1;
throughput >= 5.0 b#CommercialDecoder#fast#Server1 + 5.0 b#FreeDecoder#fast#Server2 + 2.0 b#FreeDecoder#slow#Server2 + 5.0 b#CommercialDecoder#fast#Server2 + 2.

framerate >= 20.0;

binary b#CommercialDecoder#fast#Server1, b                                                    der#fast#Server2, b#URLReader#url#Server1, b#Fi
```

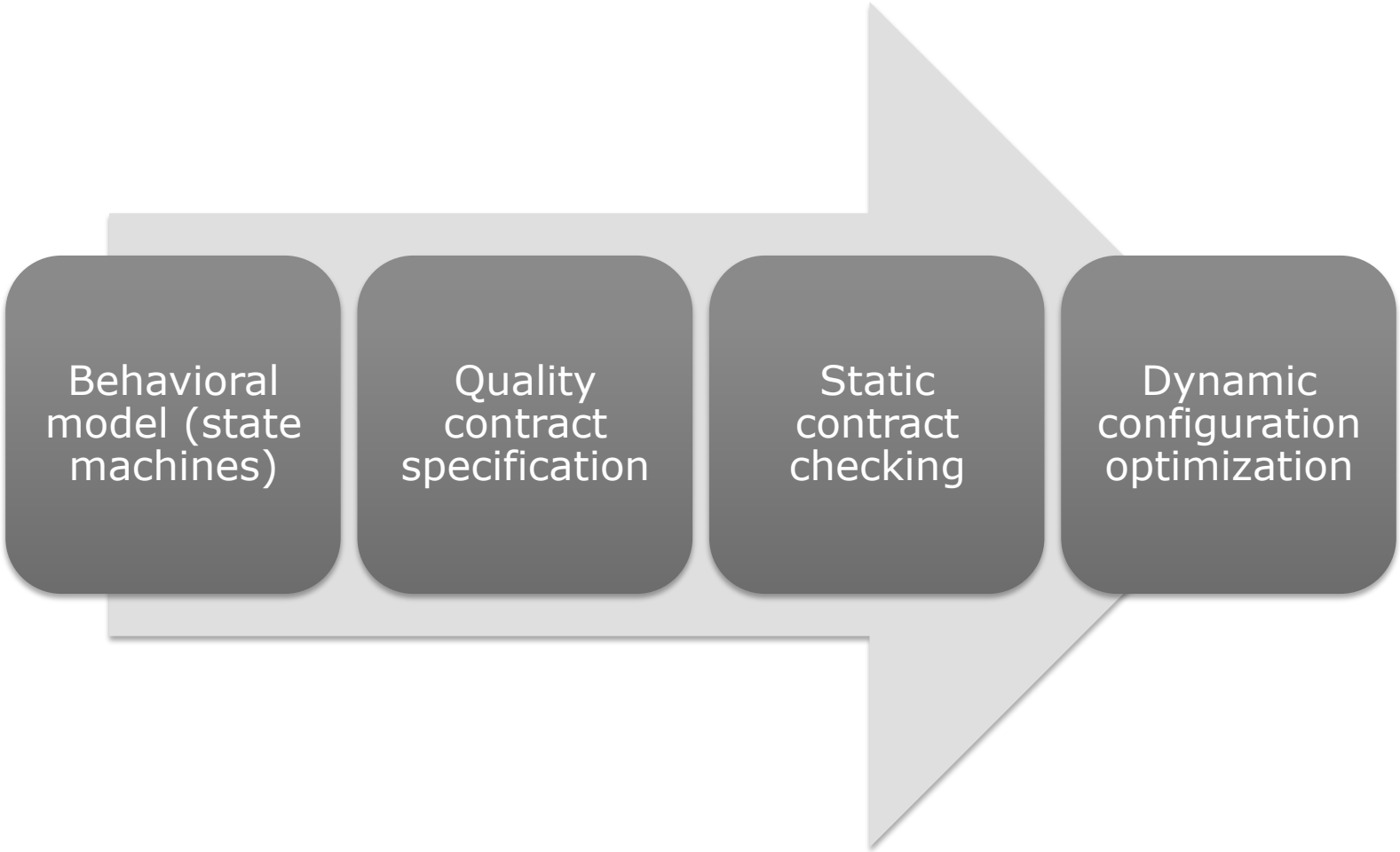


- Prices are costs
- Utility per price is an efficiency feature
- Prices can be negotated automatically
- Using a CPS will be possible with micro-pay-per-use



- All efficiency requirements can be handled




Behavioral model (state machines)
Quality contract specification
Static contract checking
Dynamic configuration optimization

TECHNISCHE UNIVERSITÄT DRESDEN

# 4.1 Seamless Interaction with CPS

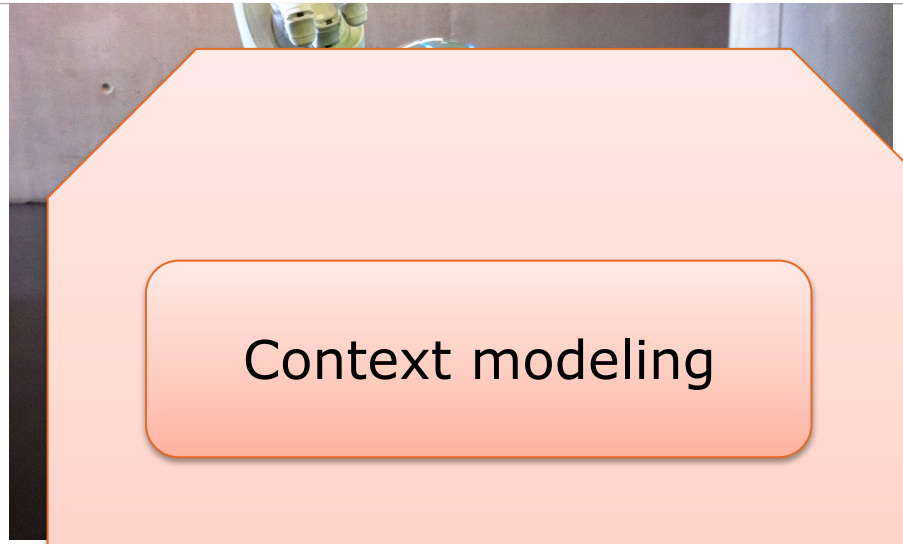
# Well, I like to teach

# But, how do I teach Chuck to bring me a coffee?



# How do I teach Chuck to bring me a coffee?

# Gesture DSL will be Domain-Specific

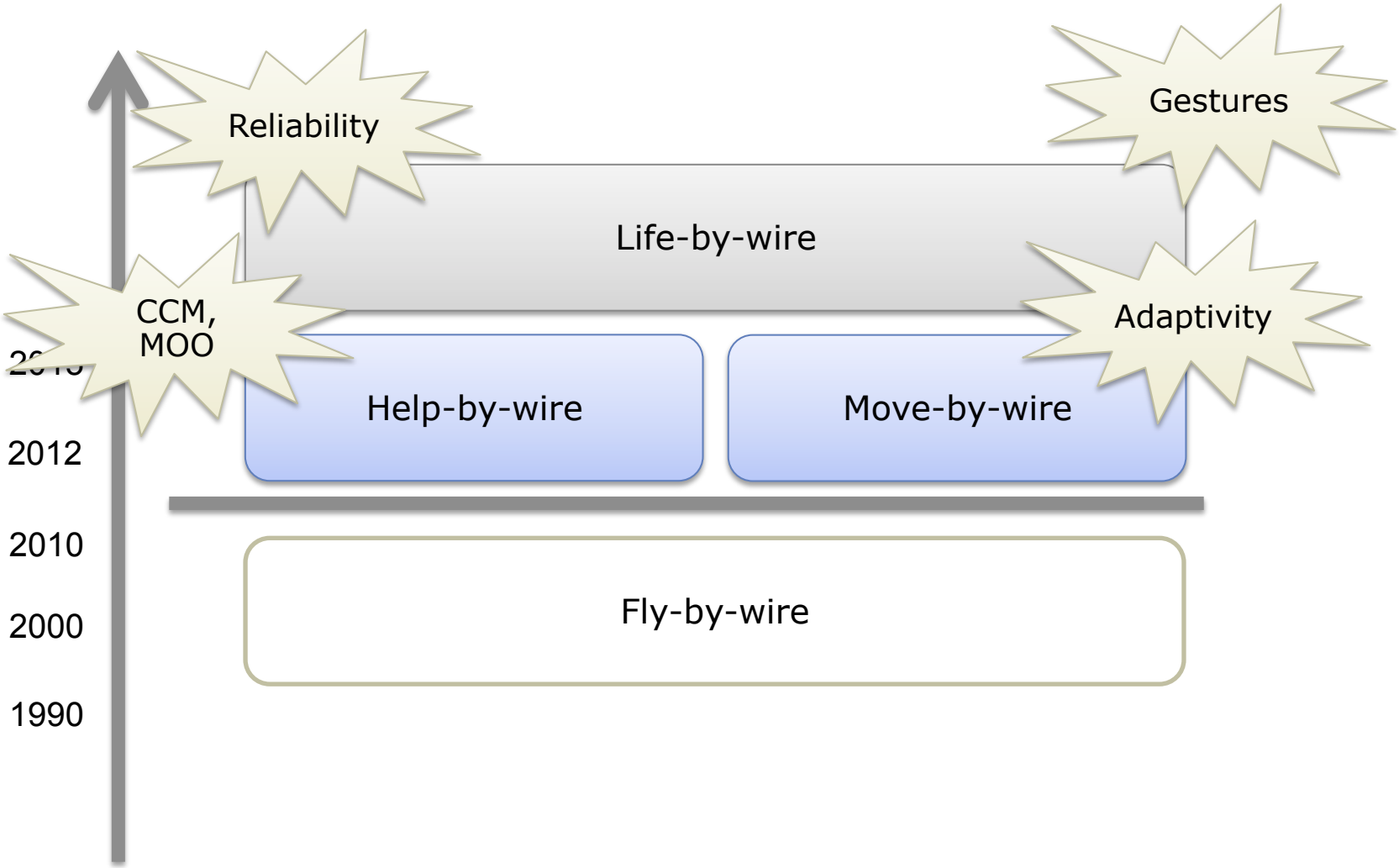


- Gestures in Cars
- Gestures in Public
- Gestures in Factories

- Context modeling
- User editable
- Semantics

TECHNISCHE UNIVERSITÄT DRESDEN

# Chap.5. Life-by-Wire


Reliability
Gestures
Life-by-wire
CCM,
MOO
Adaptivity
Help-by-wire
Move-by-wire
2012
2010
2000
1990
Fly-by-wire

- http://www.resubic.org
- Uwe.assmann@tu-dresden.de
- Andreas.richter@tu-dresden.de
- Wolfgang.lehner@tu-dresden.de
- Sebastian.goetz@tu-dresden.de



- [Acatech] Agenda CPS. Acatech Studie März 2012.