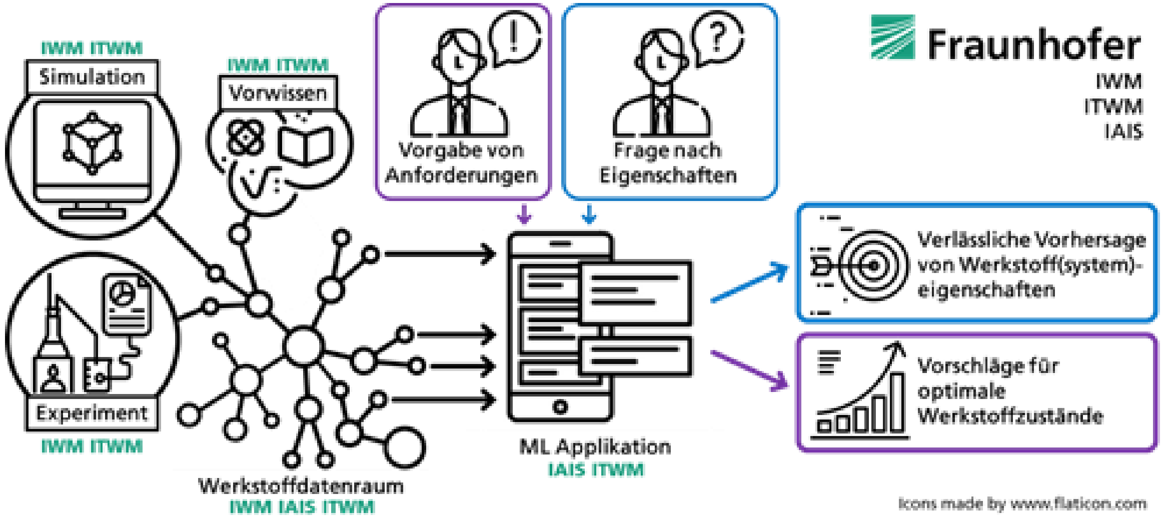
ONLINE-VERANSTALTUNG

Vorhersage von Modellparametern für die Simulation von Kabelbündeln

Abschlusskolloquium
des Fraunhofer-Konsortiums »UrWerk«
zur Entwicklung von unternehmensspezifischen Werkstoff(system)-Datenräumen



Fraunhofer
IWM
ITWM
IAIS

Icons made by www.flaticon.com

Moderation Dr. Michael Luke
Projektleiter »UrWerk«
Geschäftsfeldleiter »Bauteilsicherheit und Leichtbau«
am Fraunhofer-Institut für Werkstoffmechanik IWM

24.November 2022

Fraunhofer IWM Fraunhofer IAIS Fraunhofer ITWM

ABSCHLUSSKOLLOQUIUM „URWERK“

Vorhersage von Modellparametern für die Simulation von Kabelbündeln

Fabio Schneider-Jung

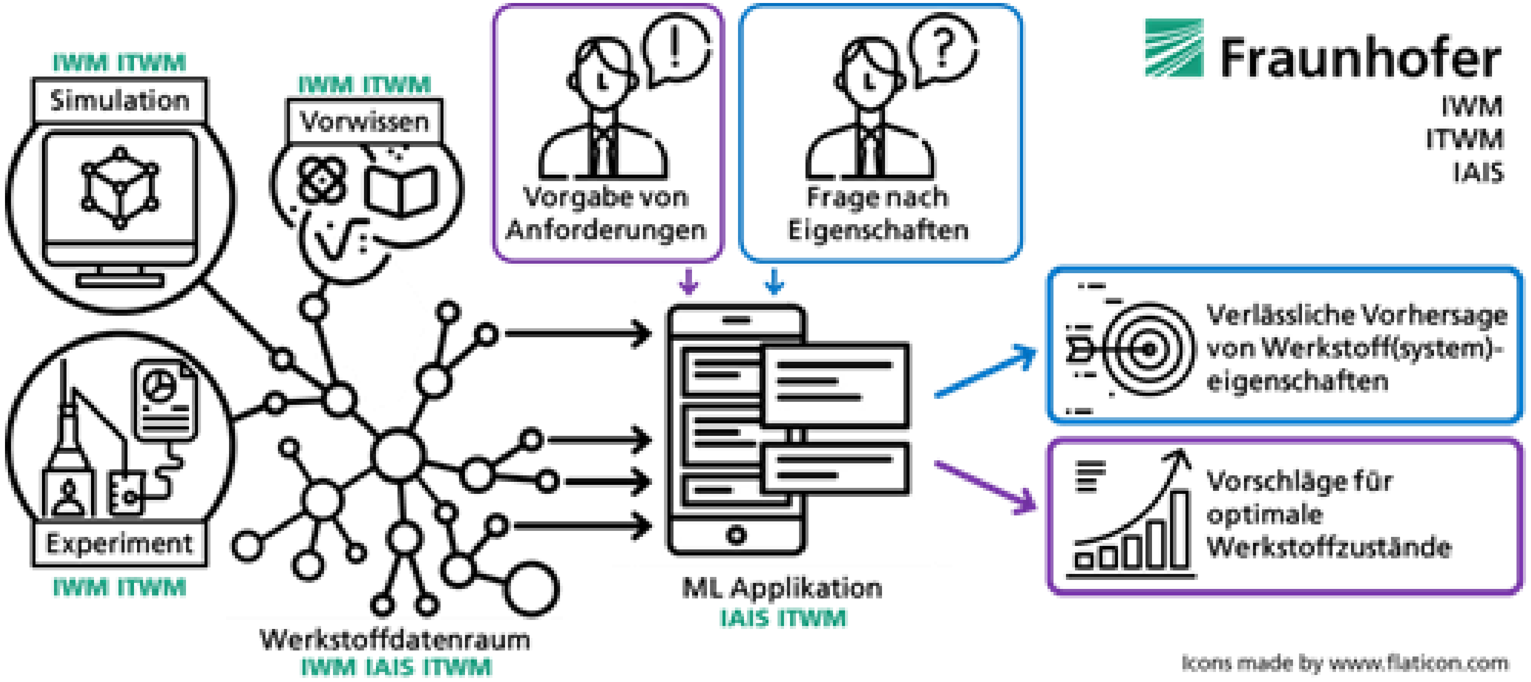
Vanessa Dörlich

Lilli Burger

Michael Burger

Joachim Linn

Dominik Jungkenn



IWM ITWM
Simulation

IWM ITWM
Vorwissen

Vorgabe von Anforderungen

Frage nach Eigenschaften

Fraunhofer
IWM
ITWM
IAIS

Experiment
IWM ITWM

Werkstoffdatenraum
IWM IAIS ITWM

ML Applikation
IAIS ITWM

Verlässliche Vorhersage von Werkstoff(system)-eigenschaften

Vorschläge für optimale Werkstoffzustände

Icons made by www.flaticon.com

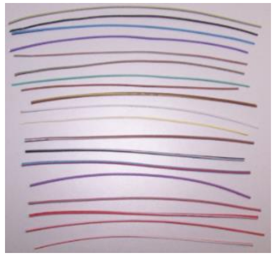
24.11.2022

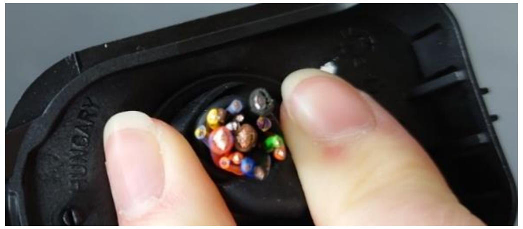
Fraunhofer IWM Fraunhofer IAIS Fraunhofer ITWM

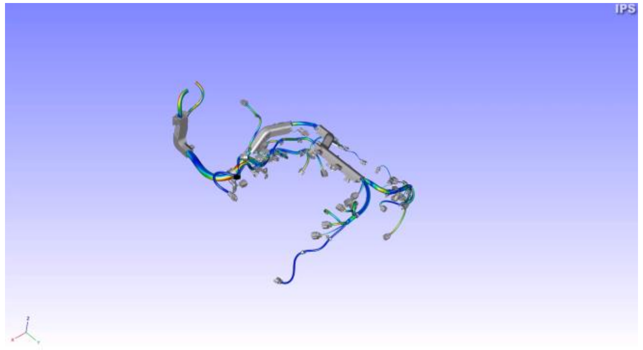
Use Case “Kabelbündel-Steifigkeiten”

Motivation

- IPS Cable Simulation
  - Quick and simple model setup & interactive simulation
  - Requires model parameters: effective stiffness
- MeSOMICS
  - Highly automated measurement system for cables and hoses
- Cable bundles
  - Cables in various combinations (depending on vehicle equipment)
  - Measurements in early phases would hinder the development







Use Case “Kabelbündel-Steifigkeiten”

Tasks in UrWerk

- Estimation of “**Werkstoffsystem**” properties: effective cable bundle stiffness

- Generate an (initial) data base by a **measurement campaign**
  - measure effective properties of cable bundles
  - measure effective properties of underlying cables

Use Case “Kabelbündel-Steifigkeiten”

Generation of data base by measurement campaign

- Generate an (initial) data base by a measurement campaign
  - measure effective properties of underlying cables

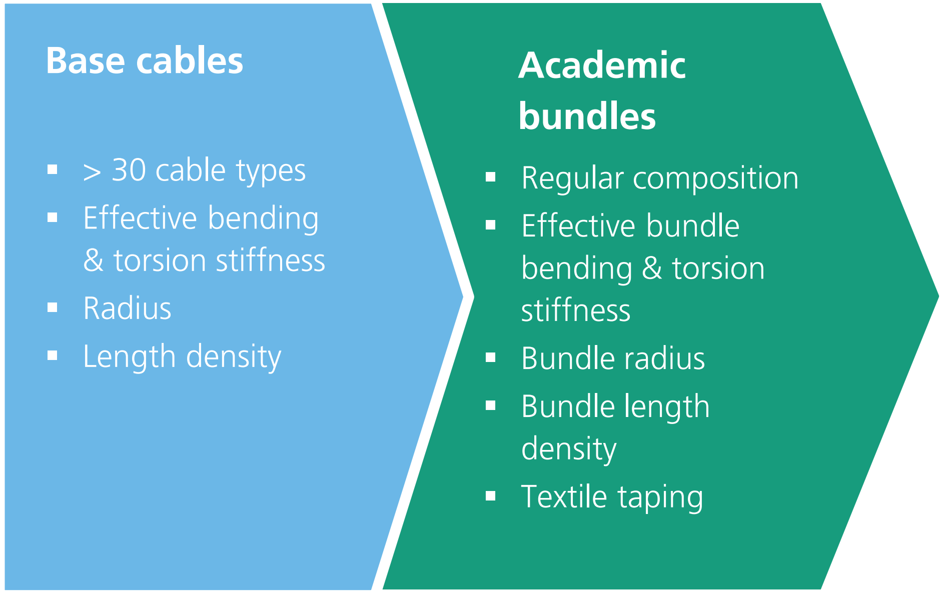
Base cables

- > 30 cable types
- Effective bending & torsion stiffness
- Radius
- Length density

Use Case “Kabelbündel-Steifigkeiten”

Generation of data base by measurement campaign

- Generate an (initial) data base by a measurement campaign
  - measure effective properties of underlying cables
  - measure effective properties of cable bundles

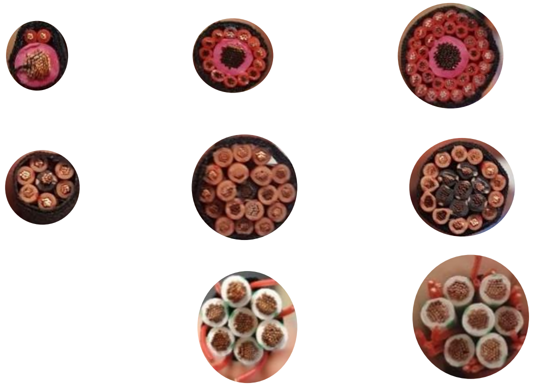


Base cables

- > 30 cable types
- Effective bending & torsion stiffness
- Radius
- Length density

Academic bundles

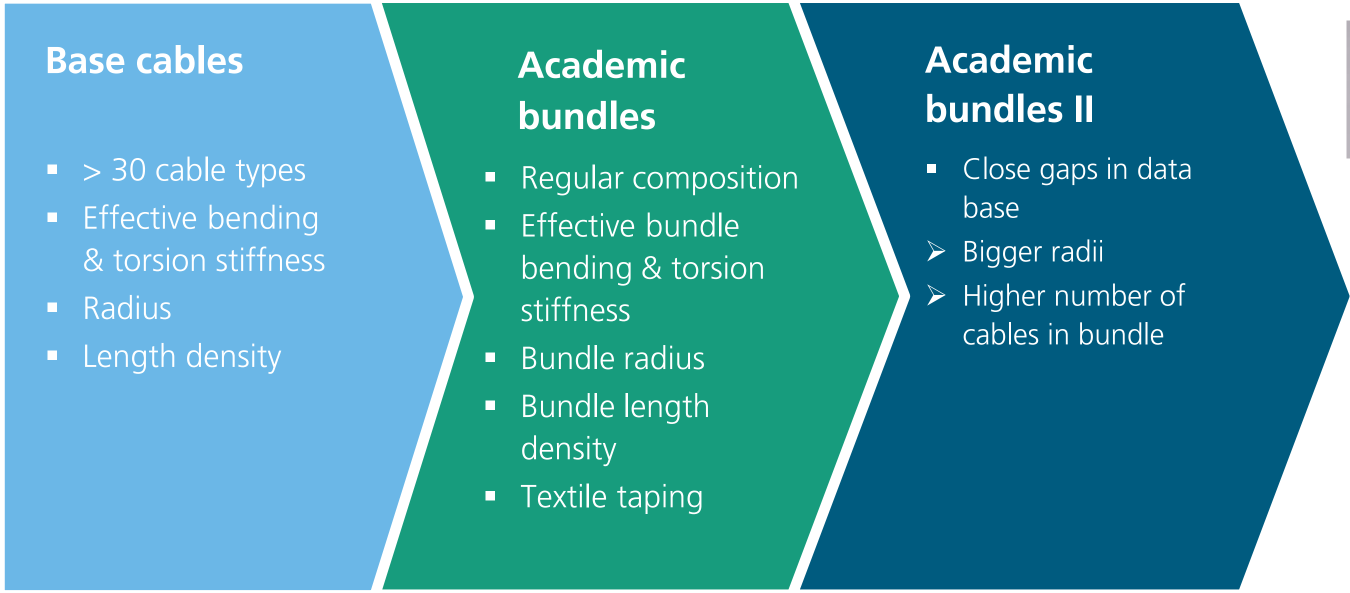
- Regular composition
- Effective bundle bending & torsion stiffness
- Bundle radius
- Bundle length density
- Textile taping



Use Case "Kabelbündel-Steifigkeiten"

Generation of data base by measurement campaign

- Generate an (initial) data base by a measurement campaign
  - measure effective properties of underlying cables
  - measure effective properties of cable bundles
- Iterative process: Measurement → Analysis → Adapt data base



Base cables

- > 30 cable types
- Effective bending & torsion stiffness
- Radius
- Length density

Academic bundles

- Regular composition
- Effective bundle bending & torsion stiffness
- Bundle radius
- Bundle length density
- Textile taping

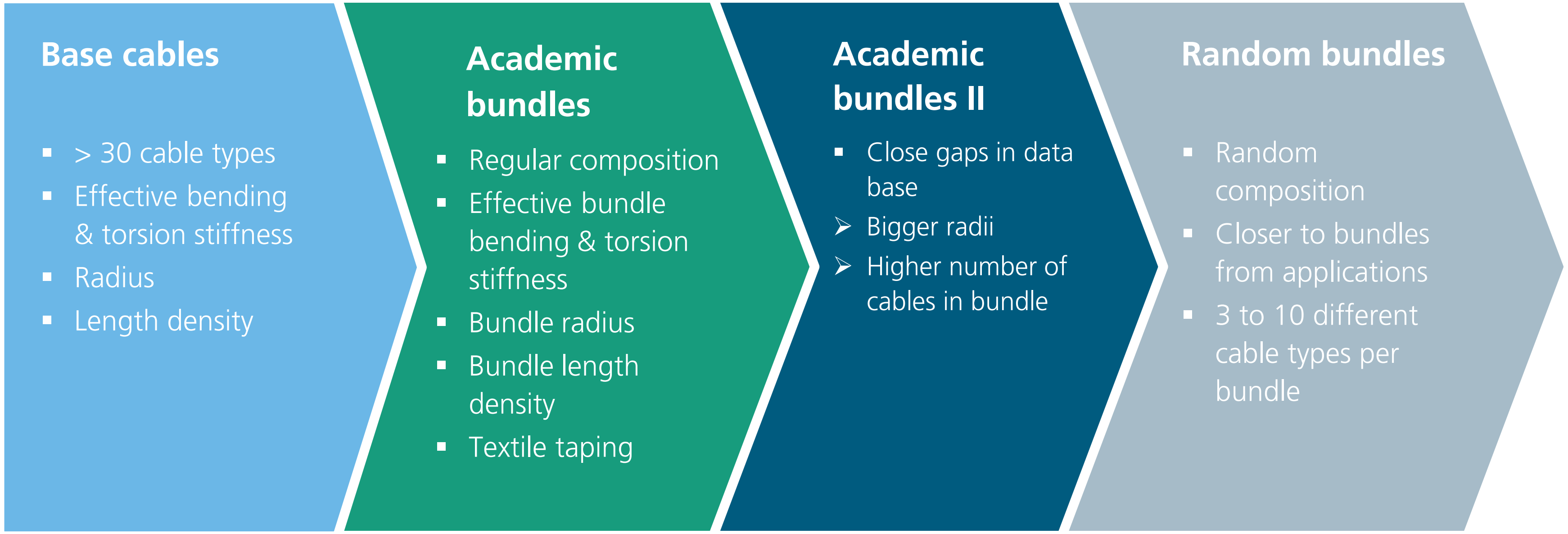
Academic bundles II

- Close gaps in data base
- ➢ Bigger radii
- ➢ Higher number of cables in bundle

Use Case “Kabelbündel-Steifigkeiten”

Generation of data base by measurement campaign

- Generate an (initial) data base by a measurement campaign
  - measure effective properties of underlying cables
  - measure effective properties of cable bundles
- Iterative process: Measurement → Analysis → Adapt data base → Validation measurements



Base cables

- > 30 cable types
- Effective bending & torsion stiffness
- Radius
- Length density

Academic bundles

- Regular composition
- Effective bundle bending & torsion stiffness
- Bundle radius
- Bundle length density
- Textile taping

Academic bundles II

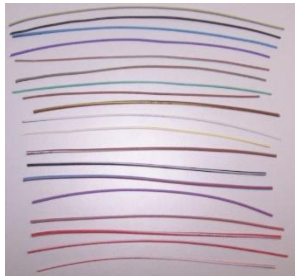
- Close gaps in data base
  - ➢ Bigger radii
  - ➢ Higher number of cables in bundle

Random bundles

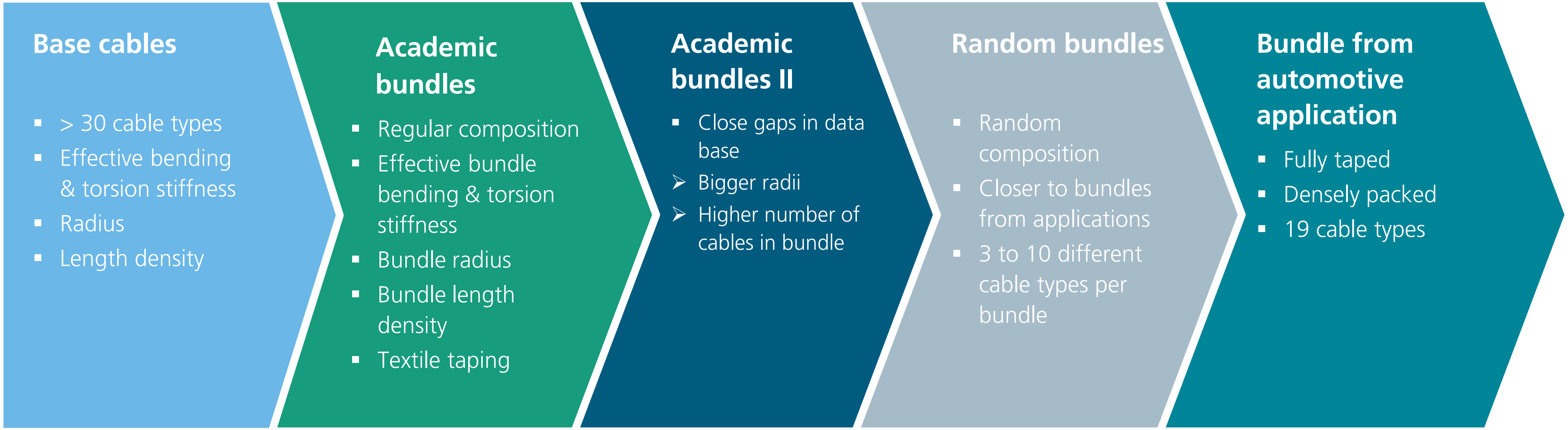
- Random composition
- Closer to bundles from applications
- 3 to 10 different cable types per bundle

Use Case “Kabelbündel-Steifigkeiten”

Generation of data base by measurement campaign



- Generate an (initial) data base by a measurement campaign
  - measure effective properties of underlying cables
  - measure effective properties of cable bundles
- Iterative process: Measurement → Analysis → Adapt data base → Validation measurements



Base cables

- > 30 cable types
- Effective bending & torsion stiffness
- Radius
- Length density

Academic bundles

- Regular composition
- Effective bundle bending & torsion stiffness
- Bundle radius
- Bundle length density
- Textile taping

Academic bundles II

- Close gaps in data base
- ➢ Bigger radii
- ➢ Higher number of cables in bundle

Random bundles

- Random composition
- Closer to bundles from applications
- 3 to 10 different cable types per bundle

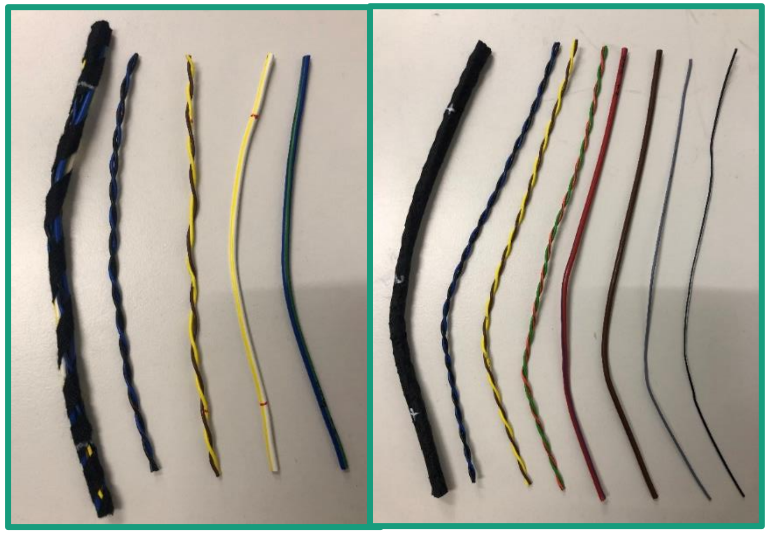
Bundle from automotive application

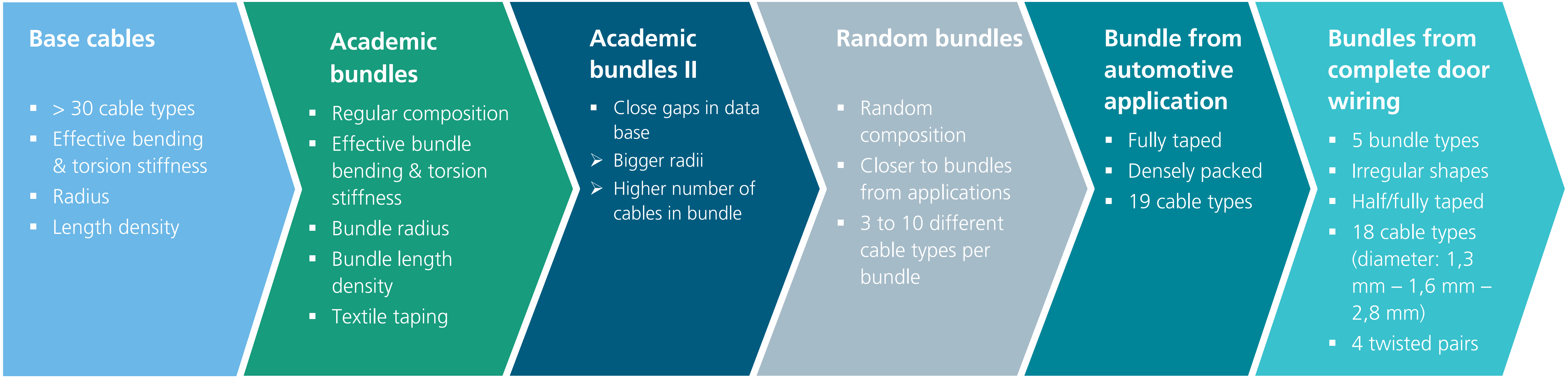
- Fully taped
- Densely packed
- 19 cable types

Use Case “Kabelbündel-Steifigkeiten”

Generation of data base by measurement campaign

- Generate an (initial) data base by a measurement campaign
  - measure effective properties of underlying cables
  - measure effective properties of cable bundles
- Iterative process: Measurement → Analysis → Adapt data base → Validation measurements





Base cables

- > 30 cable types
- Effective bending & torsion stiffness
- Radius
- Length density

Academic bundles

- Regular composition
- Effective bundle bending & torsion stiffness
- Bundle radius
- Bundle length density
- Textile taping

Academic bundles II

- Close gaps in data base
- ➢ Bigger radii
- ➢ Higher number of cables in bundle

Random bundles

- Random composition
- Closer to bundles from applications
- 3 to 10 different cable types per bundle

Bundle from automotive application

- Fully taped
- Densely packed
- 19 cable types

Bundles from complete door wiring

- 5 bundle types
- Irregular shapes
- Half/fully taped
- 18 cable types (diameter: 1,3 mm – 1,6 mm – 2,8 mm)
- 4 twisted pairs

Use Case “Kabelbündel-Steifigkeiten”

Tasks in UrWerk

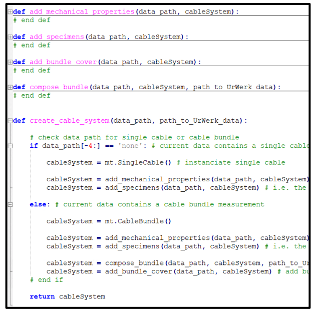
- Estimation of “**Werkstoffsystem**” properties: effective cable bundle stiffness

- Generate an (initial) data base by a **measurement campaign**
  - measure effective properties of cable bundles
  - measure effective properties of underlying cables

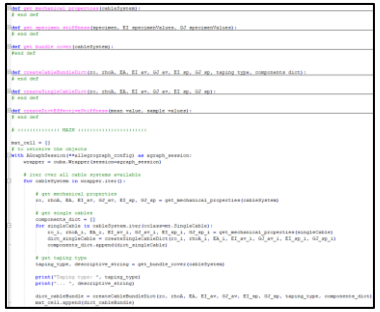
- Convenient data storage utilizing **ontological description**
  - Develop ontology for cables and cable bundles
  - → Intuitive virtual composition of bundles
  - Python scripts for up- and download

SimPhoNy

OSP-core CUDS API



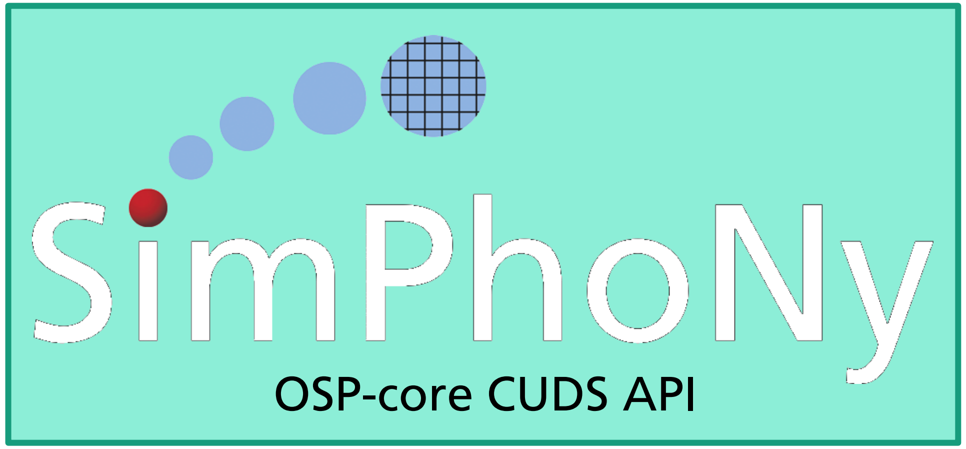
```
def add_mechanical_properties(data_path, cableSystem):
# end def

def add_specimens(data_path, cableSystem):
# end def

def add_bundle_cover(data_path, cableSystem):
# end def

def compose_bundle(data_path, cableSystem, path_to_UrWerk_data):
# end def


def create_cable_system(data_path, path_to_UrWerk_data):

    # check data path for single cable or cable bundle
    if data_path[-4:] == 'none': # current data contains a single cable

        cableSystem = mt.SingleCable() # instanciate single cable

        cableSystem = add_mechanical_properties(data_path, cableSystem)
        cableSystem = add_specimens(data_path, cableSystem) # i.e. the

    else: # current data contains a cable bundle measurement

        cableSystem = mt.CableBundle()

        cableSystem = add_mechanical_properties(data_path, cableSystem)
        cableSystem = add_specimens(data_path, cableSystem) # i.e. the

        cableSystem = compose_bundle(data_path, cableSystem, path_to_Ur
        cableSystem = add_bundle_cover(data_path, cableSystem) # add bu
    # end if

    return cableSystem
```



```
def get_mechanical_properties(cableSystem):
# end def

def get_specimen_stiffness(specimen, EI_specimenValues, GJ_specimenValues):
# end def

def get_bundle_cover(cableSystem):
#end def


def createCableBundleDict(ro, rhoA, EA, EI_av, GJ_av, EI_sp, GJ_sp, taping_type, components_dict):
# end def

def createSingleCableDict(ro, rhoA, EA, EI_av, GJ_av, EI_sp, GJ_sp):
# end def

def createDictEffectiveStiffness(mean_value, sample_values):
# end def

# :::::::::::::: MAIN ::::::::::::::::::::::

mat_cell = []
# to retreive the objects
with AGraphSession(**allegrograph_config) as agraph_session:
    wrapper = cuba.Wrapper(session=agraph_session)

    # iter over all cable systems available
    for cableSystem in wrapper.iter():

        # get mechanical properties
        ro, rhoA, EA, EI_av, GJ_av, EI_sp, GJ_sp = get_mechanical_properties(cableSystem)

        # get single cables
        components_dict = []
        for singleCable in cableSystem.iter(oclass=mt.SingleCable):
            ro_i, rhoA_i, EA_i, EI_av_i, GJ_av_i, EI_sp_i, GJ_sp_i = get_mechanical_properties(singleCable)
            dict_singleCable = createSingleCableDict(ro_i, rhoA_i, EA_i, EI_av_i, GJ_av_i, EI_sp_i, GJ_sp_i)
            components_dict.append(dict_singleCable)

        # get taping type
        taping_type, descriptive_string = get_bundle_cover(cableSystem)

        print("Taping type: ", taping_type)
        print("... ", descriptive_string)

        dict_cableBundle = createCableBundleDict(ro, rhoA, EA, EI_av, GJ_av, EI_sp, GJ_sp, taping_type, components_dict)
        mat_cell.append(dict_cableBundle)
```

Virtual composition of cable bundles

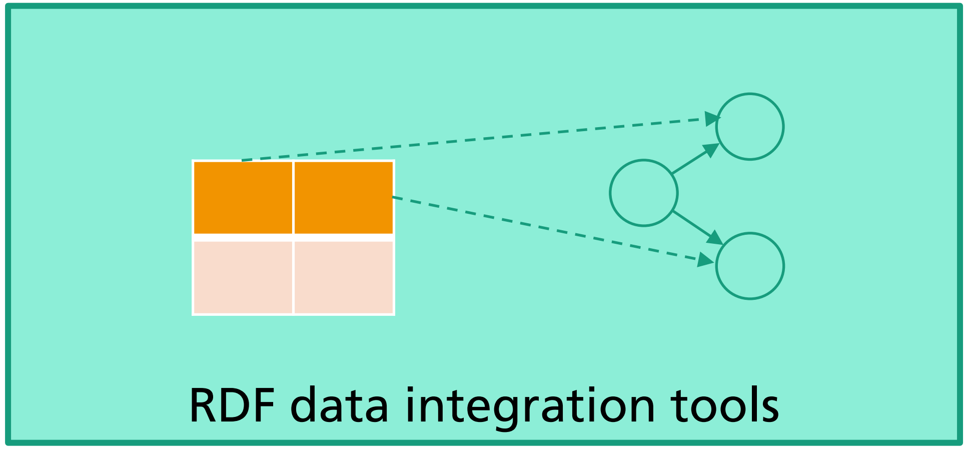
Cable Bundle use case



OSP-core CUDS API

- Very generic and flexible.
- Python interface to the ontology and data.

Fatigue use case

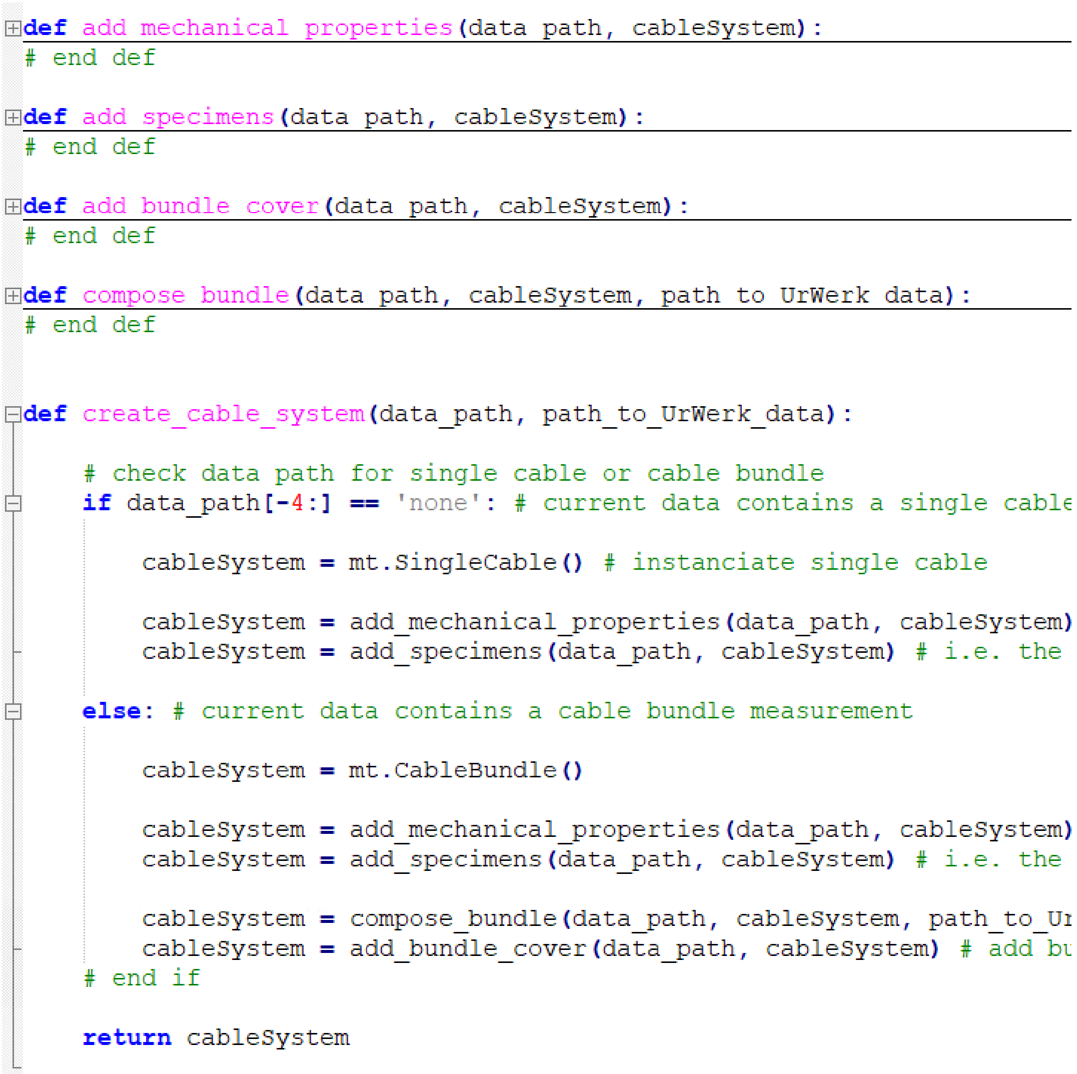


RDF data integration tools

- Not as flexible.
- More standardized.
- Closer to an "easy to use" solution (filling a template instead of actual programming).

Virtual composition of cable bundles

- *The ontology description allows storage of measured data in an intuitive, structured way*
- **Create single cable CUDS object**
  - add measured mechanical properties and single specimen data
- **Create cable bundle CUDS object**
  - add measured mechanical properties and single specimen data
  - Compose bundle **by adding cable CUDS objects**
  - and specify bundle cover („taping“ type)
- **→ virtual bundle composition**

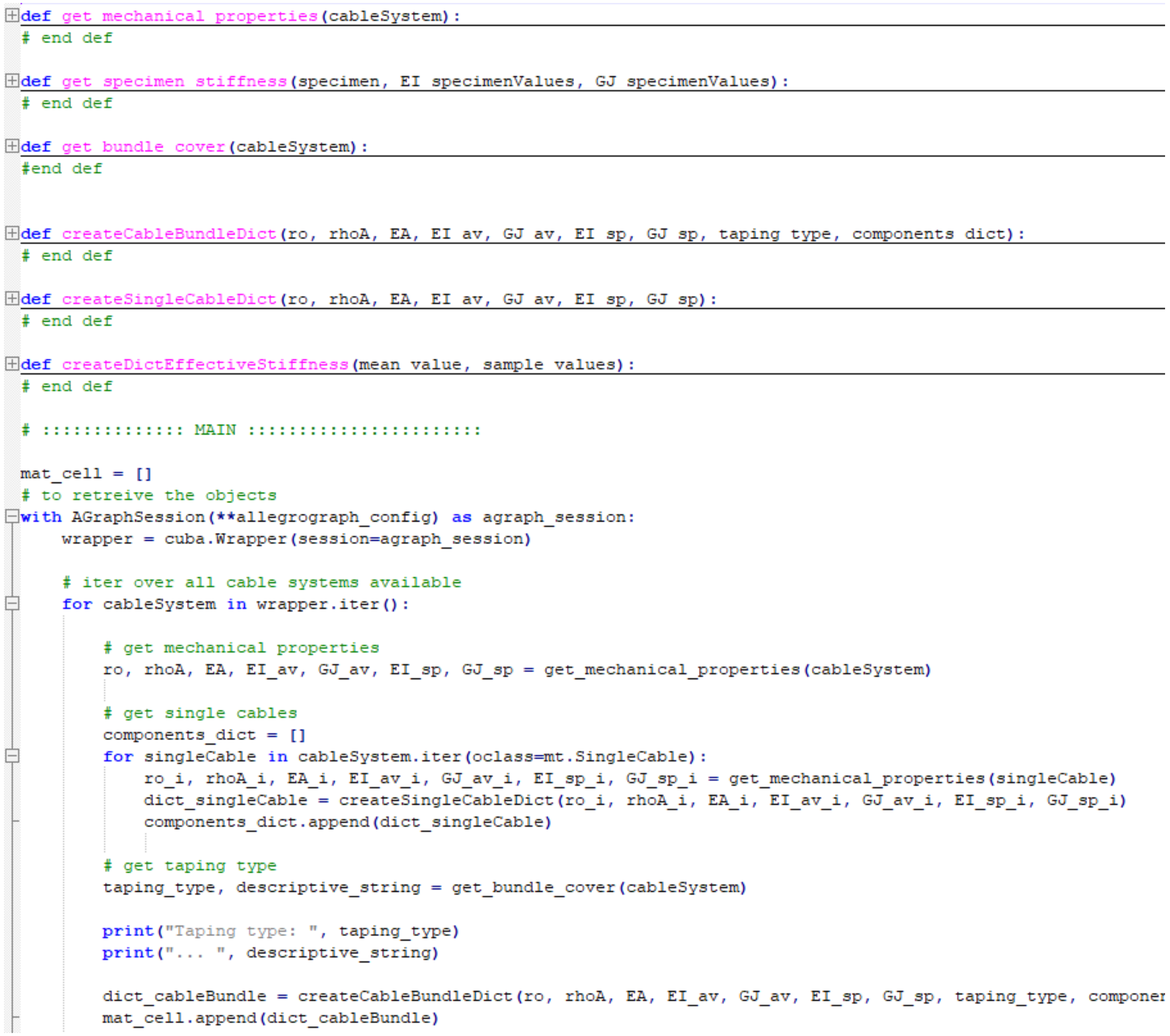


```
def add mechanical properties(data path, cableSystem):
# end def

def add specimens(data path, cableSystem):
# end def

def add bundle cover(data path, cableSystem):
# end def

def compose bundle(data path, cableSystem, path to UrWerk data):
# end def


def create_cable_system(data_path, path_to_UrWerk_data):

    # check data path for single cable or cable bundle
    if data_path[-4:] == 'none': # current data contains a single cable

        cableSystem = mt.SingleCable() # instanciate single cable

        cableSystem = add_mechanical_properties(data_path, cableSystem)
        cableSystem = add_specimens(data_path, cableSystem) # i.e. the

    else: # current data contains a cable bundle measurement

        cableSystem = mt.CableBundle()

        cableSystem = add_mechanical_properties(data_path, cableSystem)
        cableSystem = add_specimens(data_path, cableSystem) # i.e. the

        cableSystem = compose_bundle(data_path, cableSystem, path_to_Ur
        cableSystem = add_bundle_cover(data_path, cableSystem) # add bu
    # end if

    return cableSystem
```

Data download / Data query

- *The ontology description allows storage of measured data in an intuitive, structured way*

- For analysis of data, bundle and cable **data must be downloaded**
  - Python script getting back data



```
def get_mechanical_properties(cableSystem):
# end def

def get_specimen_stiffness(specimen, EI_specimenValues, GJ_specimenValues):
# end def

def get_bundle_cover(cableSystem):
#end def


def createCableBundleDict(ro, rhoA, EA, EI_av, GJ_av, EI_sp, GJ_sp, taping_type, components_dict):
# end def

def createSingleCableDict(ro, rhoA, EA, EI_av, GJ_av, EI_sp, GJ_sp):
# end def

def createDictEffectiveStiffness(mean_value, sample_values):
# end def

# :::::::::::::: MAIN :::::::::::::::::::::::

mat_cell = []
# to retreive the objects
with AGraphSession(**allegrograph_config) as agraph_session:
    wrapper = cuba.Wrapper(session=agraph_session)

    # iter over all cable systems available
    for cableSystem in wrapper.iter():

        # get mechanical properties
        ro, rhoA, EA, EI_av, GJ_av, EI_sp, GJ_sp = get_mechanical_properties(cableSystem)

        # get single cables
        components_dict = []
        for singleCable in cableSystem.iter(oclass=mt.SingleCable):
            ro_i, rhoA_i, EA_i, EI_av_i, GJ_av_i, EI_sp_i, GJ_sp_i = get_mechanical_properties(singleCable)
            dict_singleCable = createSingleCableDict(ro_i, rhoA_i, EA_i, EI_av_i, GJ_av_i, EI_sp_i, GJ_sp_i)
            components_dict.append(dict_singleCable)

        # get taping type
        taping_type, descriptive_string = get_bundle_cover(cableSystem)

        print("Taping type: ", taping_type)
        print("... ", descriptive_string)

        dict_cableBundle = createCableBundleDict(ro, rhoA, EA, EI_av, GJ_av, EI_sp, GJ_sp, taping_type, componer
        mat_cell.append(dict_cableBundle)
```

Use Case “Kabelbündel-Steifigkeiten”

Tasks in UrWerk

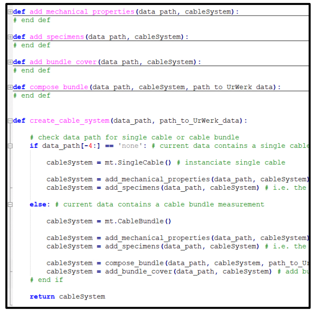
- Estimation of “**Werkstoffsystem**” properties: effective cable bundle stiffness

- Generate an (initial) data base by a **measurement campaign**
  - measure effective properties of cable bundles
  - measure effective properties of underlying cables

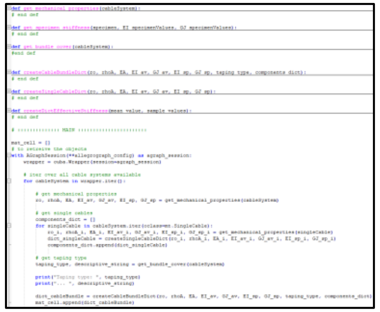
- Convenient data storage utilizing **ontological description**
  - Develop ontology for cables and cable bundles
  - → Intuitive virtual composition of bundles
  - Python scripts for up- and download

SimPhoNy

OSP-core CUDS API

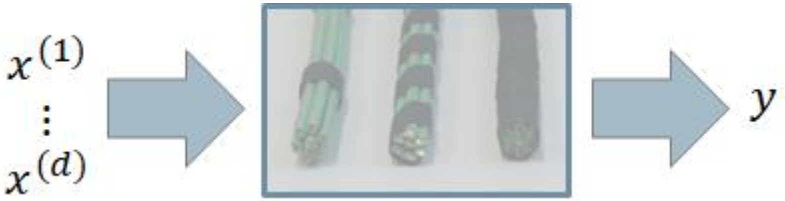


```
def add_mechanical_properties(data_path, cableSystem):
# end def

def add_specimens(data_path, cableSystem):
# end def

def add_bundle_cover(data_path, cableSystem):
# end def

def compose_bundle(data_path, cableSystem, path_to_UrWerk_data):
# end def


def create_cable_system(data_path, path_to_UrWerk_data):

    # check data path for single cable or cable bundle
    if data_path[-4:] == 'none': # current data contains a single cable

        cableSystem = mt.SingleCable() # instanciate single cable

        cableSystem = add_mechanical_properties(data_path, cableSystem)
        cableSystem = add_specimens(data_path, cableSystem) # i.e. the

    else: # current data contains a cable bundle measurement

        cableSystem = mt.CableBundle()

        cableSystem = add_mechanical_properties(data_path, cableSystem)
        cableSystem = add_specimens(data_path, cableSystem) # i.e. the

        cableSystem = compose_bundle(data_path, cableSystem, path_to_U
        cableSystem = add_bundle_cover(data_path, cableSystem) # add bu
    # end if

    return cableSystem
```



```
def get_mechanical_properties(cableSystem):
# end def

def get_specimen_stiffness(specimen, EI_specimenValues, GJ_specimenValues):
# end def

def get_bundle_cover(cableSystem):
#end def


def createCableBundleDict(ro, rhoA, EA, EI_av, GJ_av, EI_sp, GJ_sp, taping_type, components_dict):
# end def

def createSingleCableDict(ro, rhoA, EA, EI_av, GJ_av, EI_sp, GJ_sp):
# end def

def createDictEffectiveStiffness(mean_value, sample_values):
# end def

# :::::::::::::: MAIN ::::::::::::::::::::::

mat_cell = []
# to retreive the objects
with AGraphSession(**allegrograph_config) as agraph_session:
    wrapper = cuba.Wrapper(session=agraph_session)

    # iter over all cable systems available
    for cableSystem in wrapper.iter():

        # get mechanical properties
        ro, rhoA, EA, EI_av, GJ_av, EI_sp, GJ_sp = get_mechanical_properties(cableSystem)

        # get single cables
        components_dict = []
        for singleCable in cableSystem.iter(oclass=mt.SingleCable):
            ro_i, rhoA_i, EA_i, EI_av_i, GJ_av_i, EI_sp_i, GJ_sp_i = get_mechanical_properties(singleCable)
            dict_singleCable = createSingleCableDict(ro_i, rhoA_i, EA_i, EI_av_i, GJ_av_i, EI_sp_i, GJ_sp_i)
            components_dict.append(dict_singleCable)

        # get taping type
        taping_type, descriptive_string = get_bundle_cover(cableSystem)

        print("Taping type: ", taping_type)
        print("... ", descriptive_string)

        dict_cableBundle = createCableBundleDict(ro, rhoA, EA, EI_av, GJ_av, EI_sp, GJ_sp, taping_type, components_dict)
        mat_cell.append(dict_cableBundle)
```

- Develop, analyze and train a suitable **ML algorithm** for the
  - estimation of effective cable bundle properties (y)
  - based on given effective cable properties ($x^{(1)}, \dots, x^{(d)}$)



Vielen Dank für Ihre Aufmerksamkeit!

Kontakt

Lilli Burger, Vanessa Dörlich, Fabio Schneider-Jung

Fraunhofer-Institut für Techno- und Wirtschaftsmathematik (ITWM)
Bereich *Mathematik für die Fahrzeugentwicklung*

[Lilli.Burger, Vanessa.Doerlich, Fabio.Julian.Schneider-Jung] @itwm.fraunhofer.de