

PR1ME: A SIMULATION FRAMEWORK FOR HYDROPROCESSING

César G. Pernalete, Pieter L. Janssens, Kevin M. Van Geem, Joris W. Thybaut

Laboratory for Chemical Technology

hydroprocessing

heavy petroleum fractions ($\sim 350\text{ }^{\circ}\text{C} - 550\text{ }^{\circ}\text{C}$) are converted into valuable lighter products

Vacuum Gas Oil (VGO)



catalytic conversion

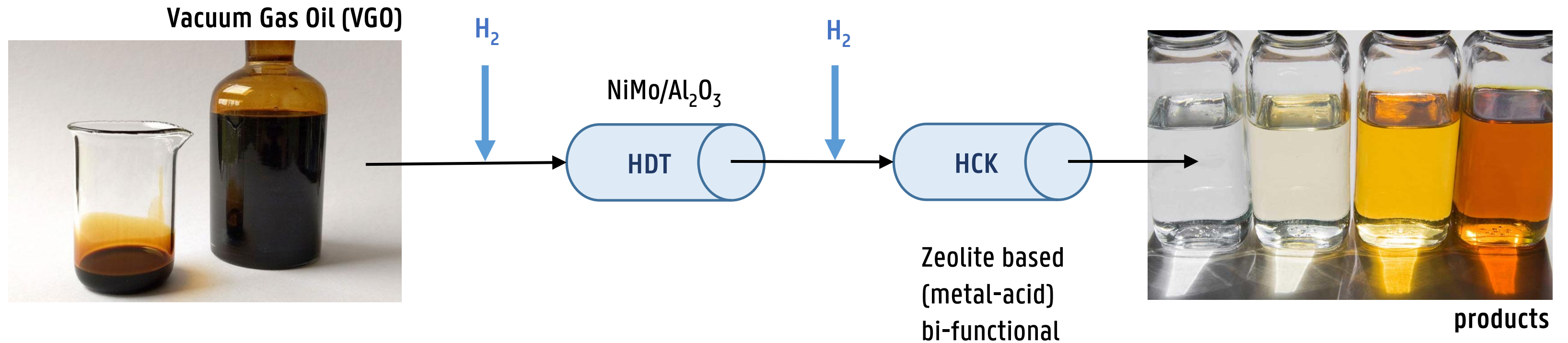


products

- naphtha
- jet fuel
- diesel
- lube oils

hydroprocessing

one or two-stage process: hydrotreating (HDT) and hydrocracking (HCK)



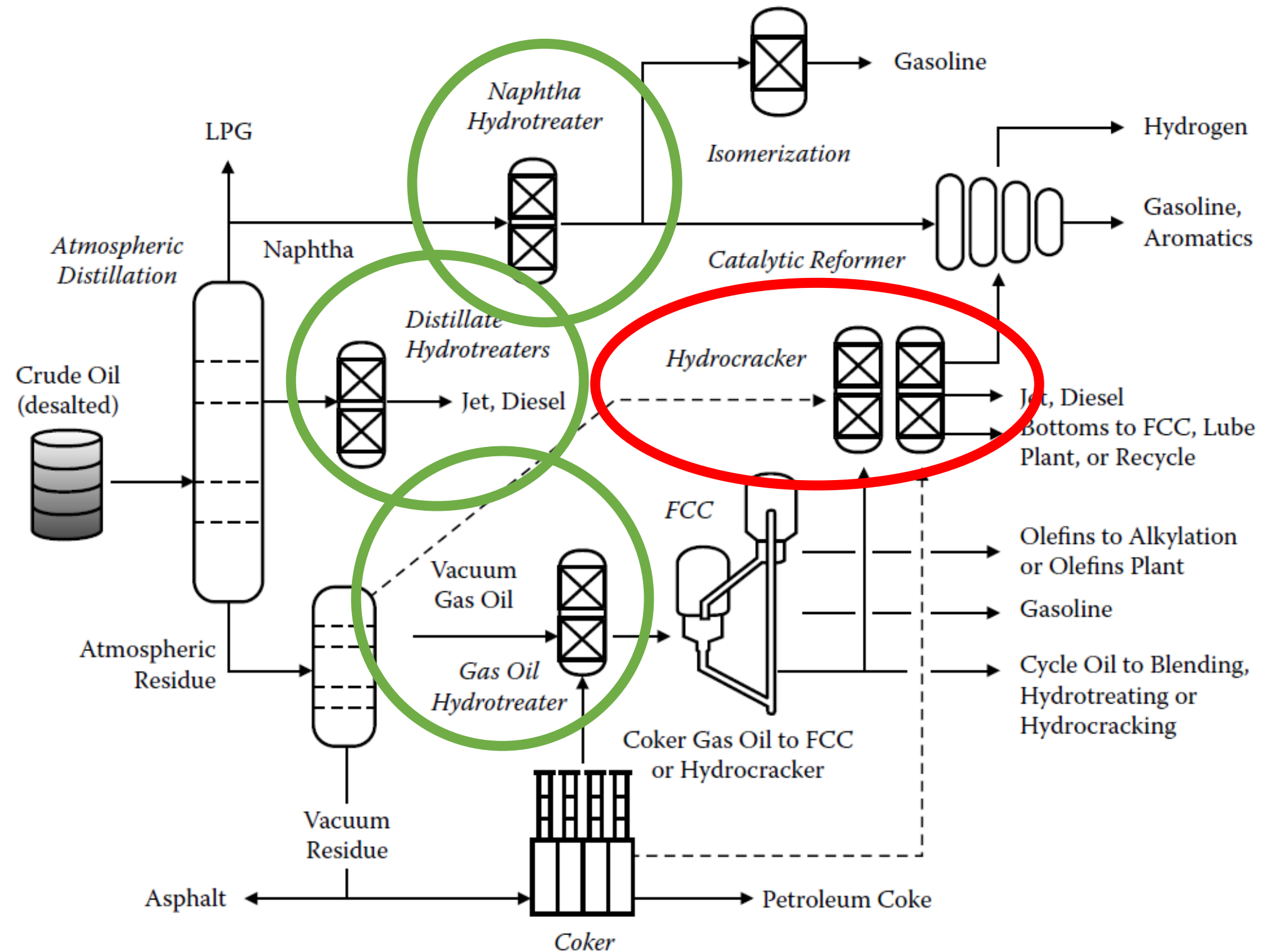
- naphtha
- jet fuel
- diesel
- lube oils

hydroprocessing in a refinery

- almost all streams in a refinery are hydroprocessed at some stage

- **HDT**: upgrading

- **HCK**: conversion



VG0 hydroprocessing

- at industrial scale feed and product streams are characterized in terms of bulk properties
- True Boiling Point (TBP) curve, P(I)ONA, etc.

Vacuum Gas Oil (VGO)



H₂

NiMo/Al₂O₃

HDT

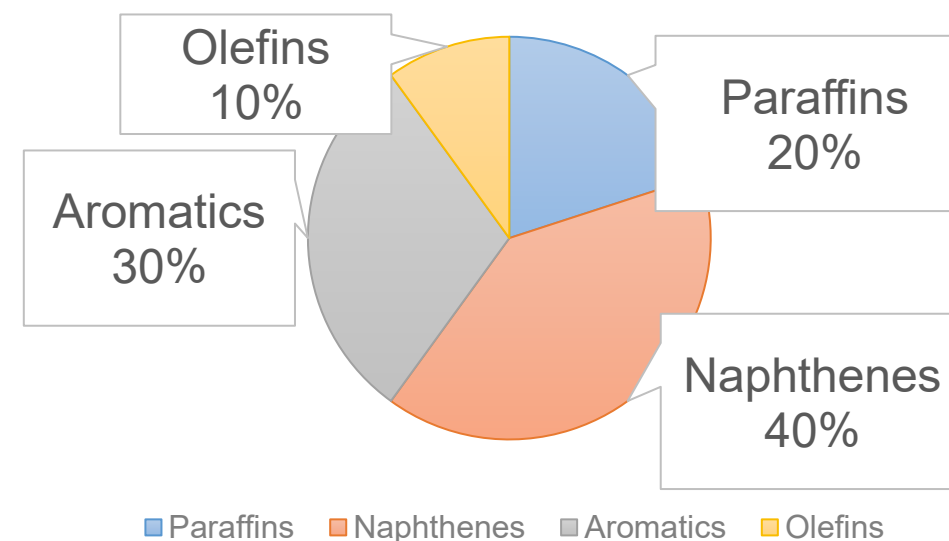
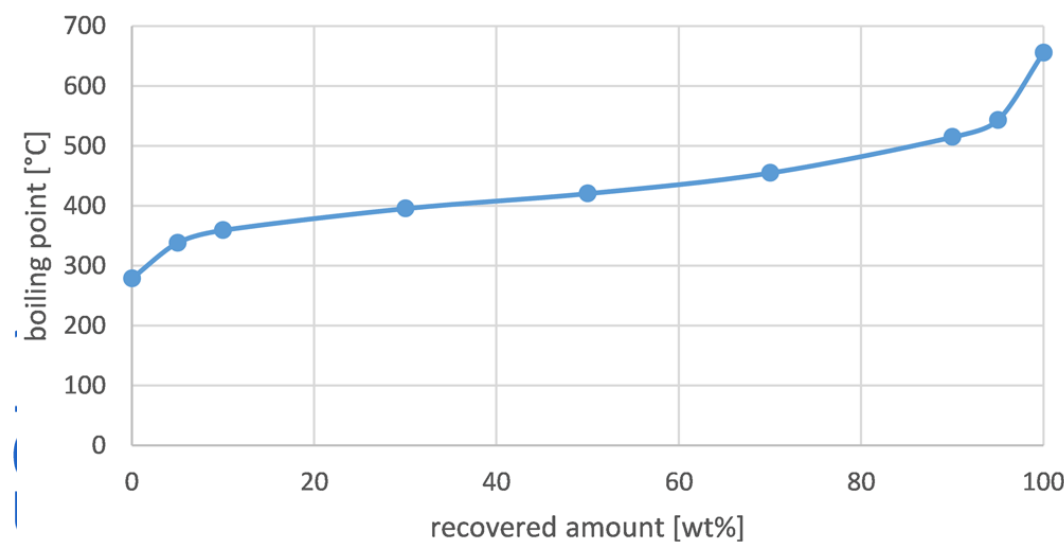
H₂

HCK

Zeolite based
(metal-acid)
bi-functional



products



- naphtha
- jet fuel
- diesel
- lube oils

outline

- Bulk properties and the Petroleum Fractions Interface
- PR1ME: from bulk properties to fundamental kinetics
- integration with PME through Petro Fractions: proof of concept
- future plans

bulk properties and the Petroleum Fractions Interface

ICapeThermoPetroleumFractions

- Handles petroleum mixtures within a Material Object.
- Enables UO to read and modify petroleum-specific properties:
 - Dist. curves, PNA, cloud point, ... (bulk properties)
- Enables run-time re-characterization of fraction properties:
 - (NBP, MW, SG, .. etc.).
- Currently supported by AVEVA™ PRO/II™

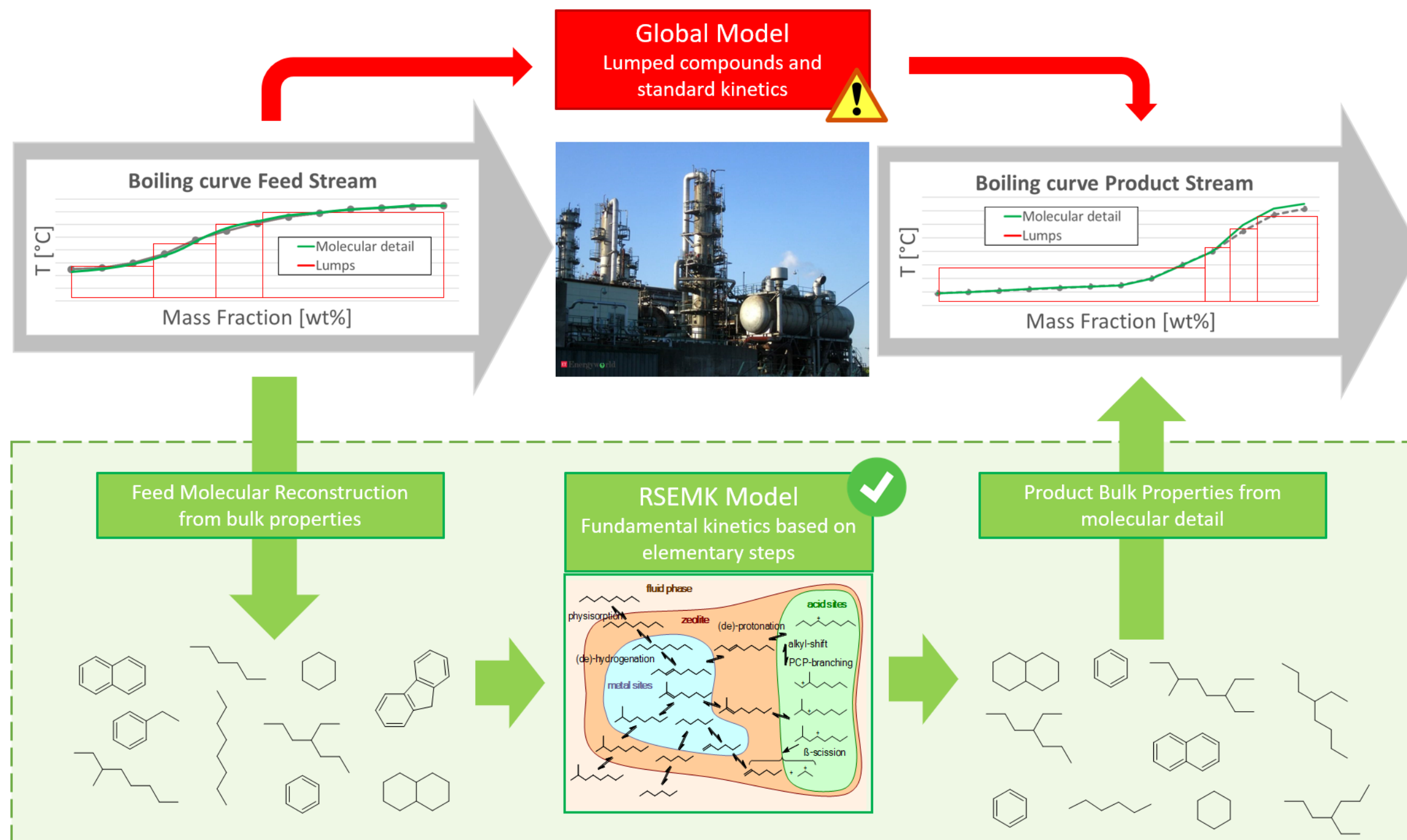


First release - 2003

Second release - 2023

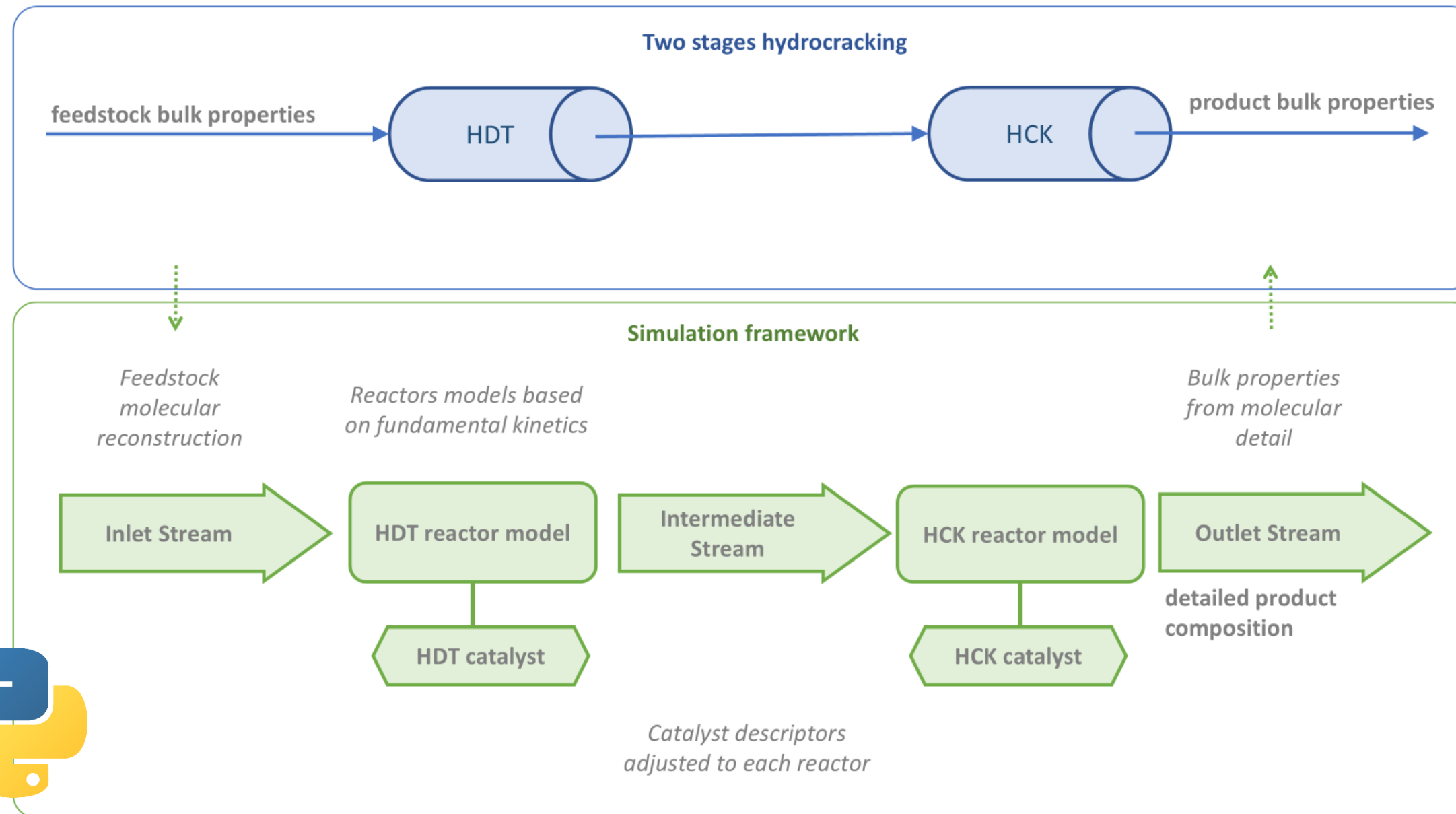
PR1ME: from bulk properties to fundamental kinetics

PR1ME integrates a Single-Event MicroKinetic model with a molecular reconstruction engine



PR1ME is a python framework

- Object-oriented architecture allows to create a combined objects flowsheet-wise

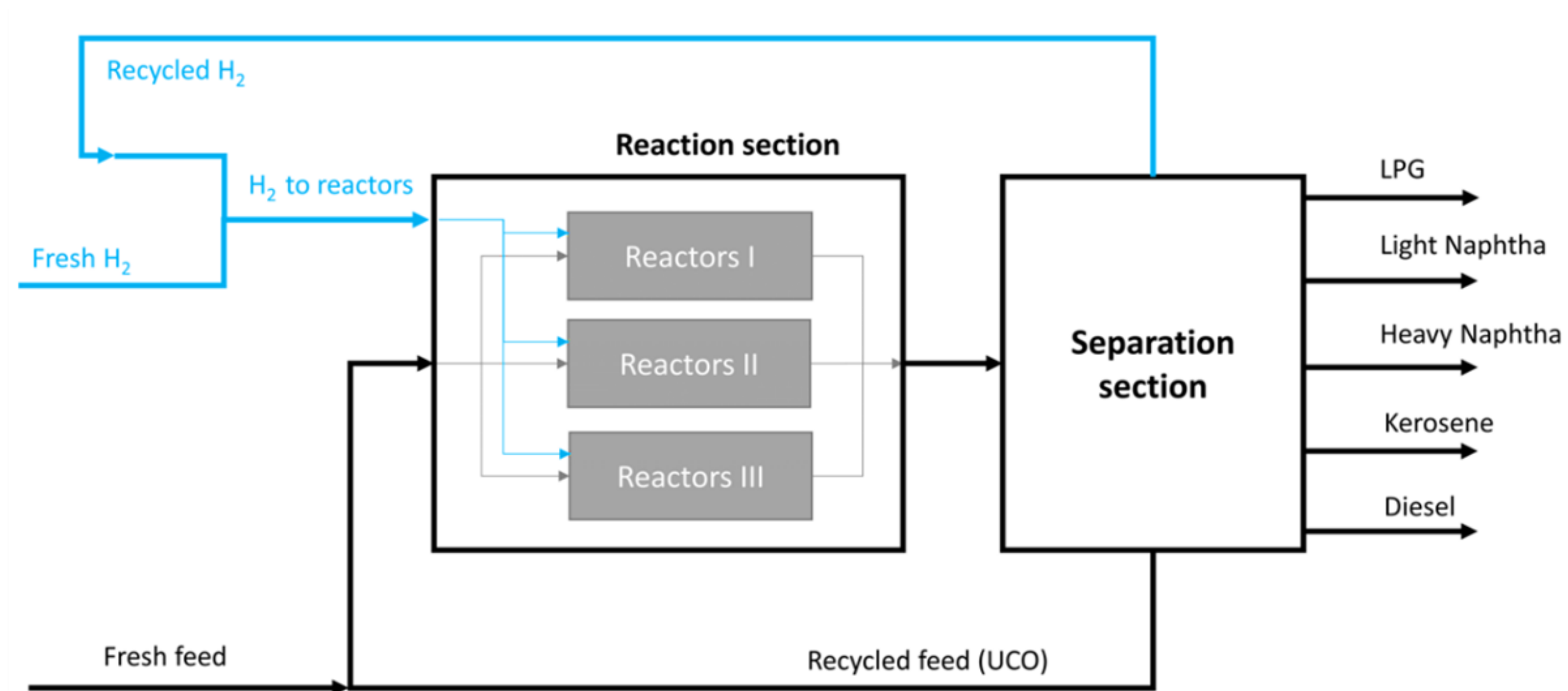


```
>>  
>> import pr1me  
>>
```



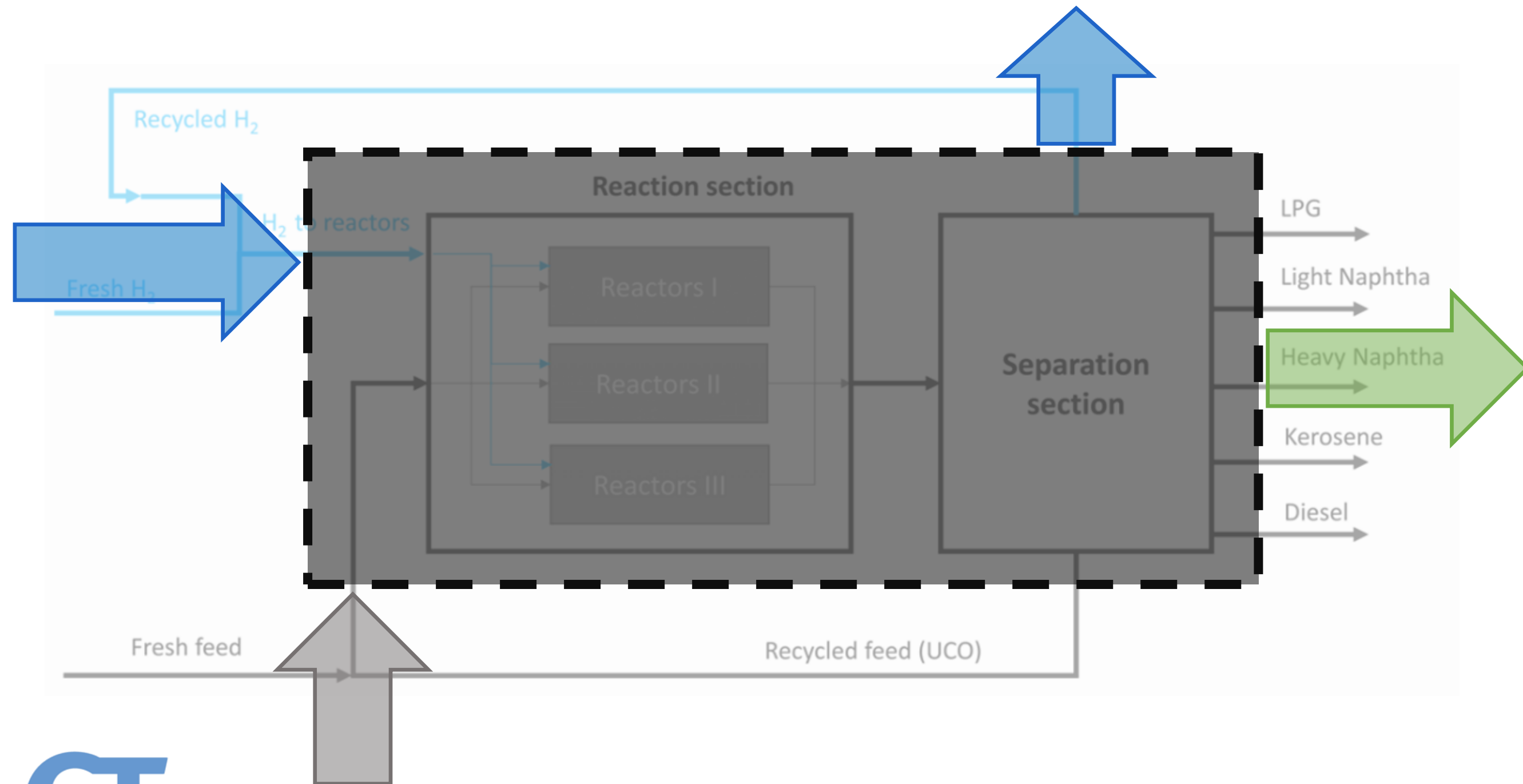
VG0 hydroprocessing at industrial scale

A configuration of three parallel downflow trickle-bed reactors with recycle was simulated



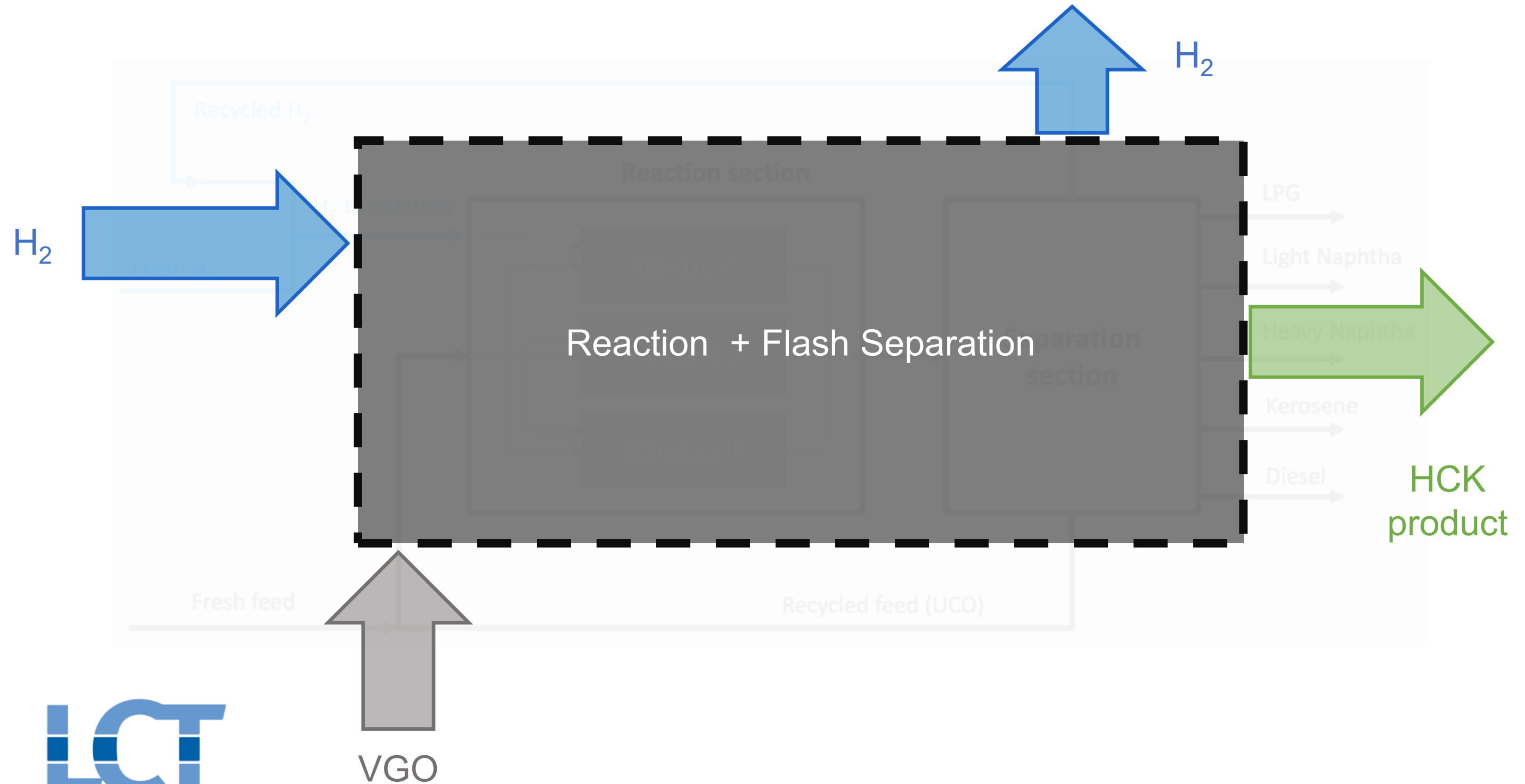
VG0 hydroprocessing at industrial scale

A simplified process scheme was selected for a proof-of-concept of the integration with the PME



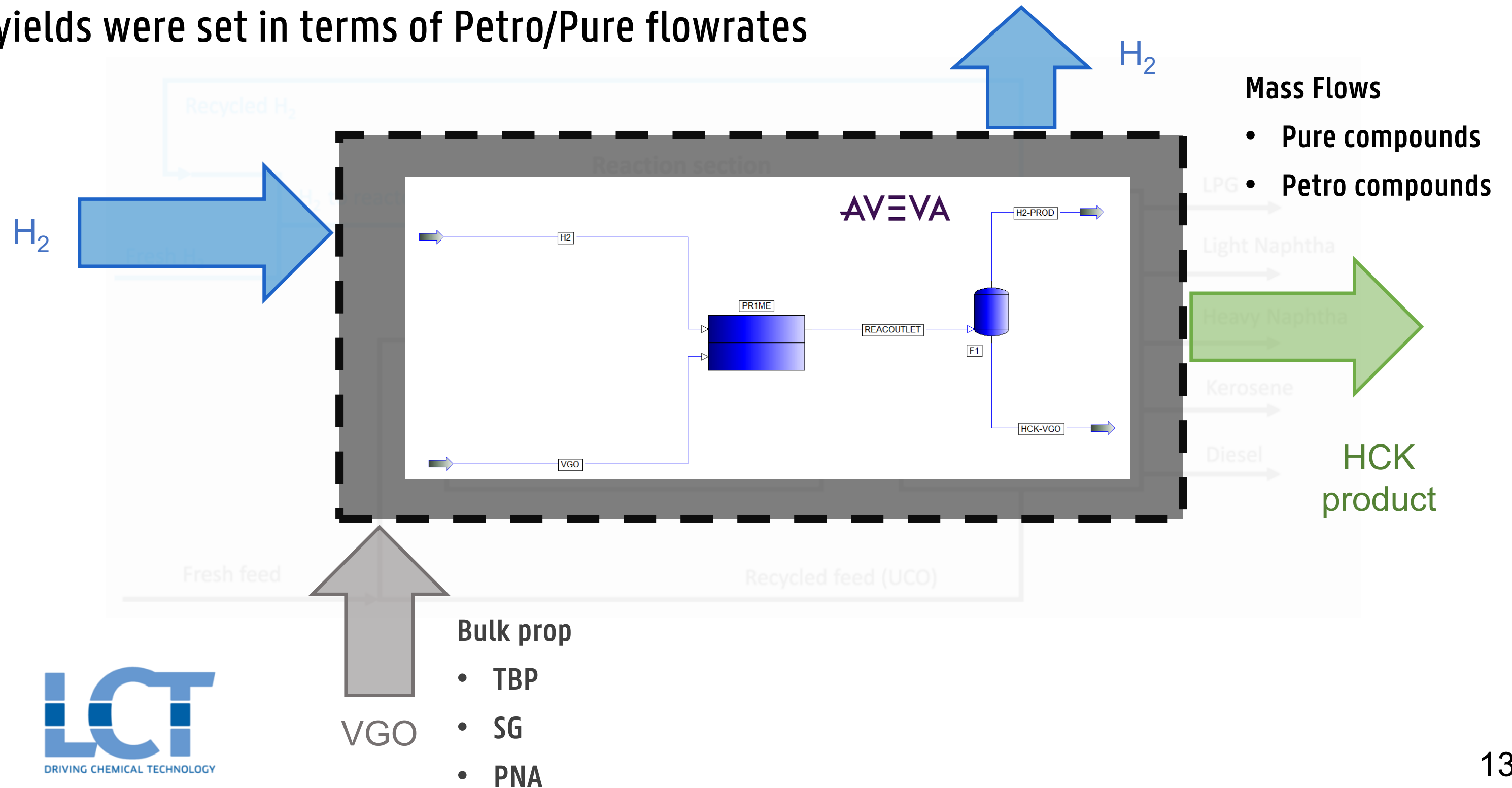
VG0 hydroprocessing at industrial scale

A simplified process scheme was selected for a proof-of-concept of the integration with the PME



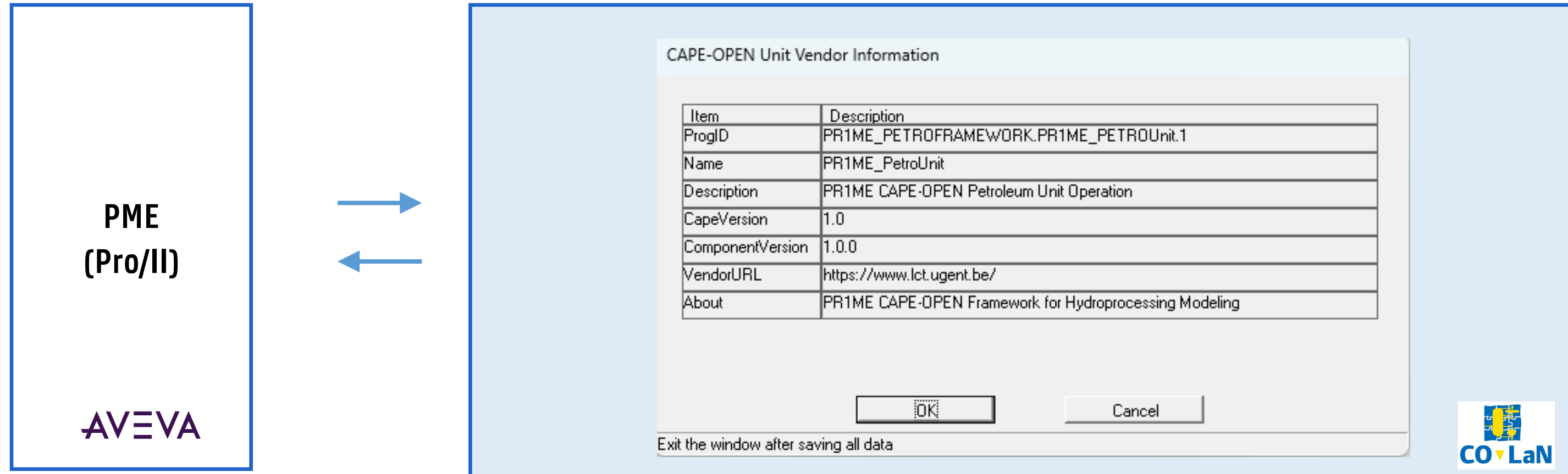
VGO hydroprocessing at industrial scale

- The feed was modeled in terms of bulk properties
- The yields were set in terms of Petro/Pure flowrates



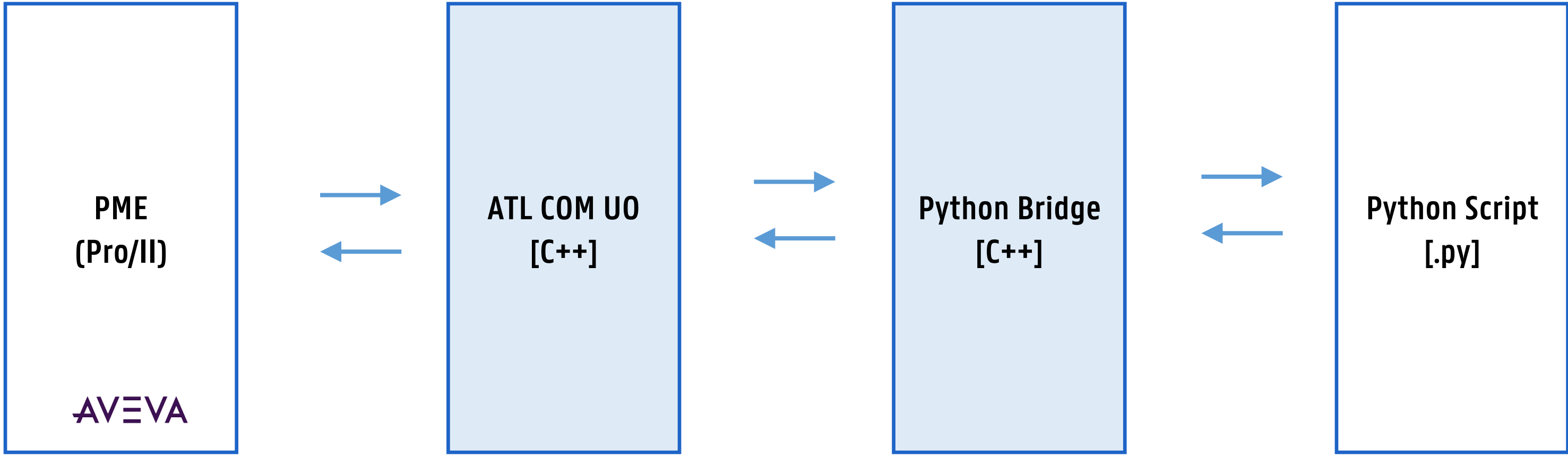
integration with PME through Petro Fractions: POC

- An ATL COM Unit Operation was implemented including ICapeThermoPetroleumFractions.



integration with PME through Petro Fractions: POC

- The ATL COM Unit Operation uses ICapeThermoPetroleumFractions Via QueryInterface



ICapeThermoMaterial
ICapeThermoPetroFractionsII

ICapeUnit

Python C API

pr1me.py

integration with PME through Petro Fractions: POC

- Communication with the PME requires specific protocol

ATL COM UO [C++]

PME	»»»	1 Query interfaces	<code>pFeedMO = ICapeThermoMaterialObject feedPF = QI(pFeedMO, ICapeThermoPetroFractionsII) prodPF = QI(pProdMO, ICapeThermoPetroFractionsII)</code>
	»»»	2 Align feed/product characterization	<code>prodPF → CopyPetroProperties(pFeedMO)</code>
	»»»	3 Read Petro Bulk inputs	<code>feedPF → GetPetroBulkProp(keysIn)</code>
	«««	4 Set overall property	<code>prodPF → SetOverallProp(keysOut, valsOut)</code>
	»»»	5 Flash Product	<code>eqProd → CalcEquilibrium()</code>
	«««	6 Write Petro bulk results	<code>prodPF → SetPetroBulkProp(keysOut, valsOut)</code>
	»»»	7 PME completes data if required	<code>prodPF → CompletePetroProperties()</code>

native C++ COM bridge to Python

- The ATL DLL embeds a “bridge” that lets native C++ COM code call Python functions at runtime.
 - Python C API

CPR1ME_PetroUnit

- ATL COM UO [C++]
- Builds JSON bulk

JSON spec



PythonHost

- C++ class
- Call(bulk.dump())

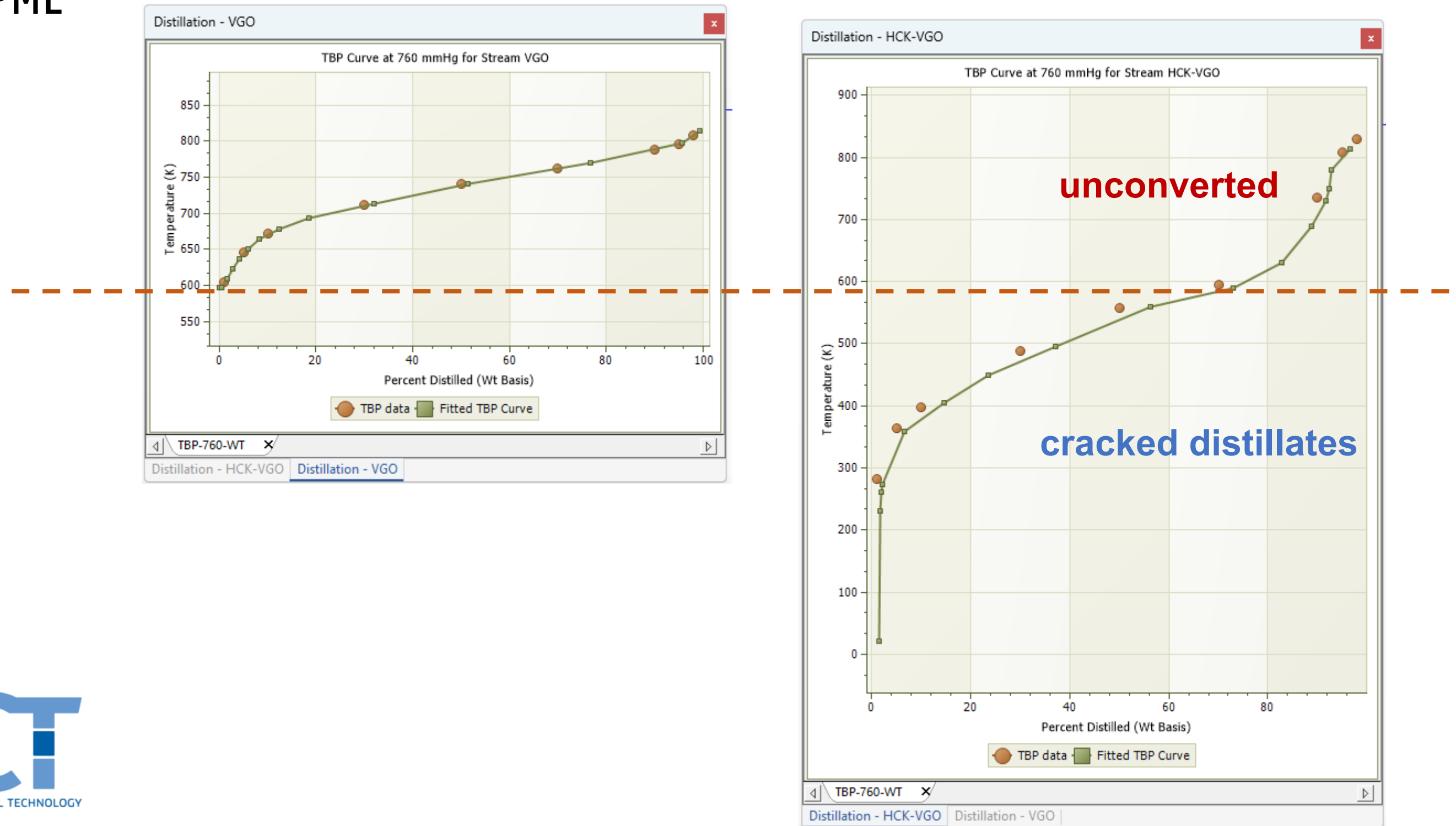


my_model.py

```
import prime
def run(bulk: str) -> str:
    r = prime.reactor.calculate()
    return json.dumps(r)
```

Proof of concept: TBP curve of feed and products

- The integration allows to monitor TBP (bulk prop.) in feed and products
 - ..with the PME



future plans

- Extend towards additional feed / product bulk properties
 - Detailed PIONA, S, N, Conradson Carbon,...etc
- Test further integration with the PME
 - recycles, sequential beds, quenching
- CAPE OPEN certification

acknowledgments

Krishna Penukonda (AVEVA)

Jasper van Baten (Amsterchem)

LABORATORY FOR CHEMICAL TECHNOLOGY

Technologiepark 125, 9052 Ghent, Belgium

E info.lct@ugent.be

T 003293311757

<https://www.lct.ugent.be>

