DEON DIGITAL

» *Contract languages, distributed ledgers, and its connection to AI* «

DEON DIGITAL

VALUE
PROPOSITION

Disruptive technologies like **Blockchain, AI, and the Internet of things (IoT)** enable the creation of new cross-industry ecosystems and will empower the machine-to-machine economy

Our **Deon Digital platform** empowers new ecosystems with

- Design and execution of digital contracts in a common and easy-to-use business language
- Seamless integration of existing legacy systems
- Multichain-integration across existing and new blockchain platforms
- Connection to the Internet of things (IoT)
- Highly efficient (functional) language with formal semantics and verification
- Highly flexible reporting and analytics capabilities

Deon's technology delivers **proven value** to clients by

- Providing over 15X faster development for decentralized transaction systems
- Enabling agile and business-led system changes
- Dramatically reducing cost of maintenance of IT systems
- Improving system soundness through failure-resistant and rigorously verified code

Our **technology is ideally suited to be combined with deep learning**

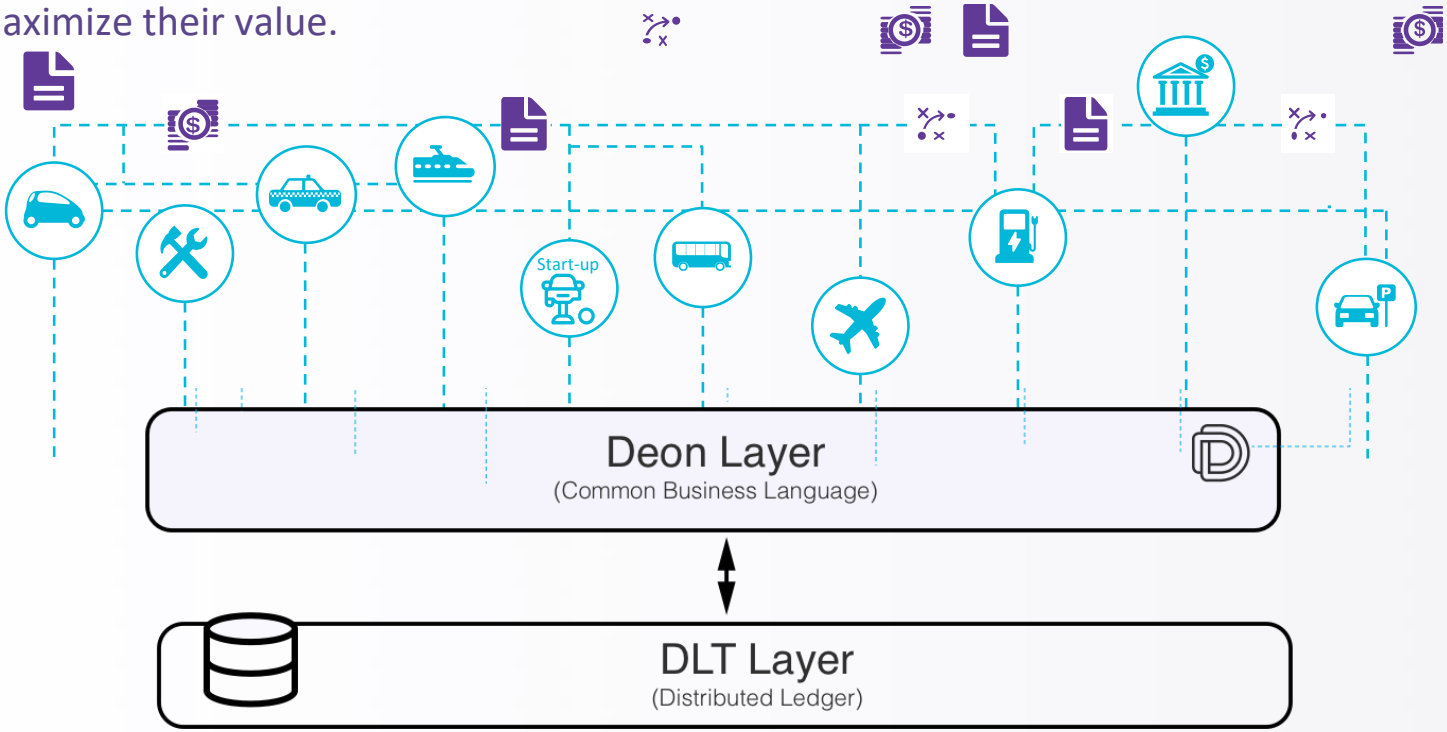
- The contract specification language has formal semantics, i.e. can interpreted like mathematics
- The embedded processes in the language span a decision tree which can used to find the best optimal decisions (actions). The reporting languages give each tree an economic value
- Large contracts or contract portfolio have too vast/deep trees to be fully enumerated
- Deep learning is used to approximate the trees and find the next best decision analogous to technics used in AlphaGo ©.

DEON DIGITAL

VALUE
PROPOSITION

Digital contracts, distributed ledger, and AI technology are a game changer! The machine-to-machine economy needs digtal contracts

Any future ecosystem is a complex network of business collaboration, which exchanges data and contracts. Such a system needs common standards, languages and decentralized technical enablers. All these machines need to constantly optimize their economic value and thus, need to make online decisions to maximize their value.



DEON DIGITAL

Idea

Provide a specific (programming) language for expressing (requirements of) the business.

... but ensure that the specifications of requirements are:

1. executable, and
2. analyzable, and
3. readable, and
4. monitorable, and
5. automatable.

"The specification is the implementation“

“The specification implies an optimal execution strategy”

(Deon) Digital “Contracts”

*"readable" specification of **future allowed and required actions** of participants in contract, business processes, or business interaction*

readable - by both machine and people

formal semantics – fully translatable to mathematical equations

specification - does not act on it's own

actions - "ought-to-do" specifications

participants - who are obligated

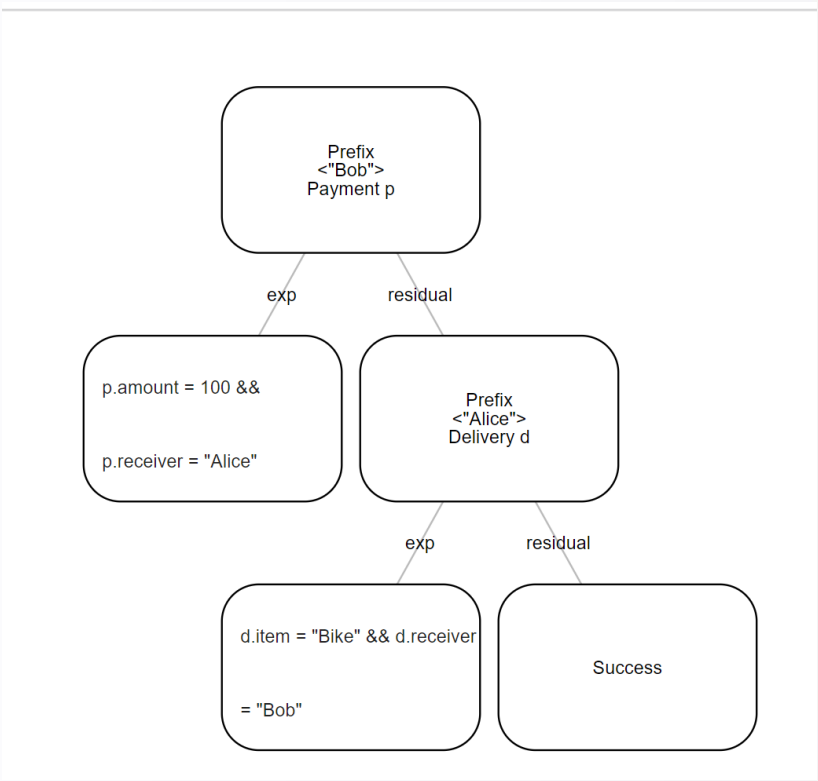
Simple example of payment-vs-bike contract/process

```
type Payment: Event {
  amount: Int,
  receiver: String }

type Delivery: Event {
  receiver: String,
  item: String}

template BikeSale(seller, buyer, amount, item) =
  <buyer> p:Payment
    where p.amount = amount && p.receiver = seller
  then
  <seller> d:Delivery
    where d.item = item  && d.receiver = buyer
  then
  success

template Main() = BikeSale("Alice","Bob",100,"Bike")
```



Simple example of payment-vs-bike contract walk through the example

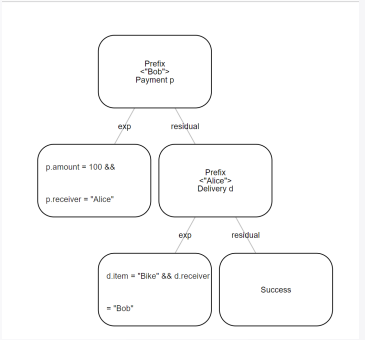
1) Instantiated contract:
   <"Bob"> p : Payment where p.amount = 100 && p.receiver = "Alice“ then <"Alice"> d : Delivery where d.item = "Bike" && d.receiver = "Bob" then success

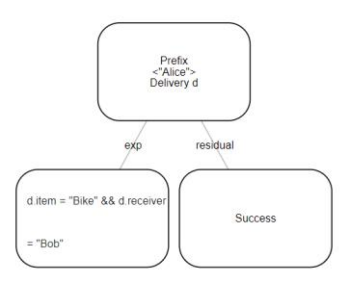
2) Event:
   [Payment { agent = "Bob", timestamp = #2018-07-02T13:35:54.567Z#, amount = 100, receiver = "Alice" }]

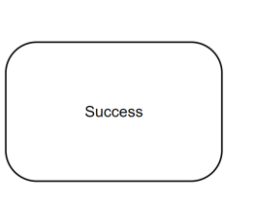
3) Residual contract:
   <"Alice"> d : Delivery where d.item = "Bike" && d.receiver = "Bob" then success

4) Event:
   Delivery { agent = "Alice", timestamp = #2018-07-02T13:38:22.132Z#, receiver = "Bob", item = "Bike" }]

5) Residual contract:
   success

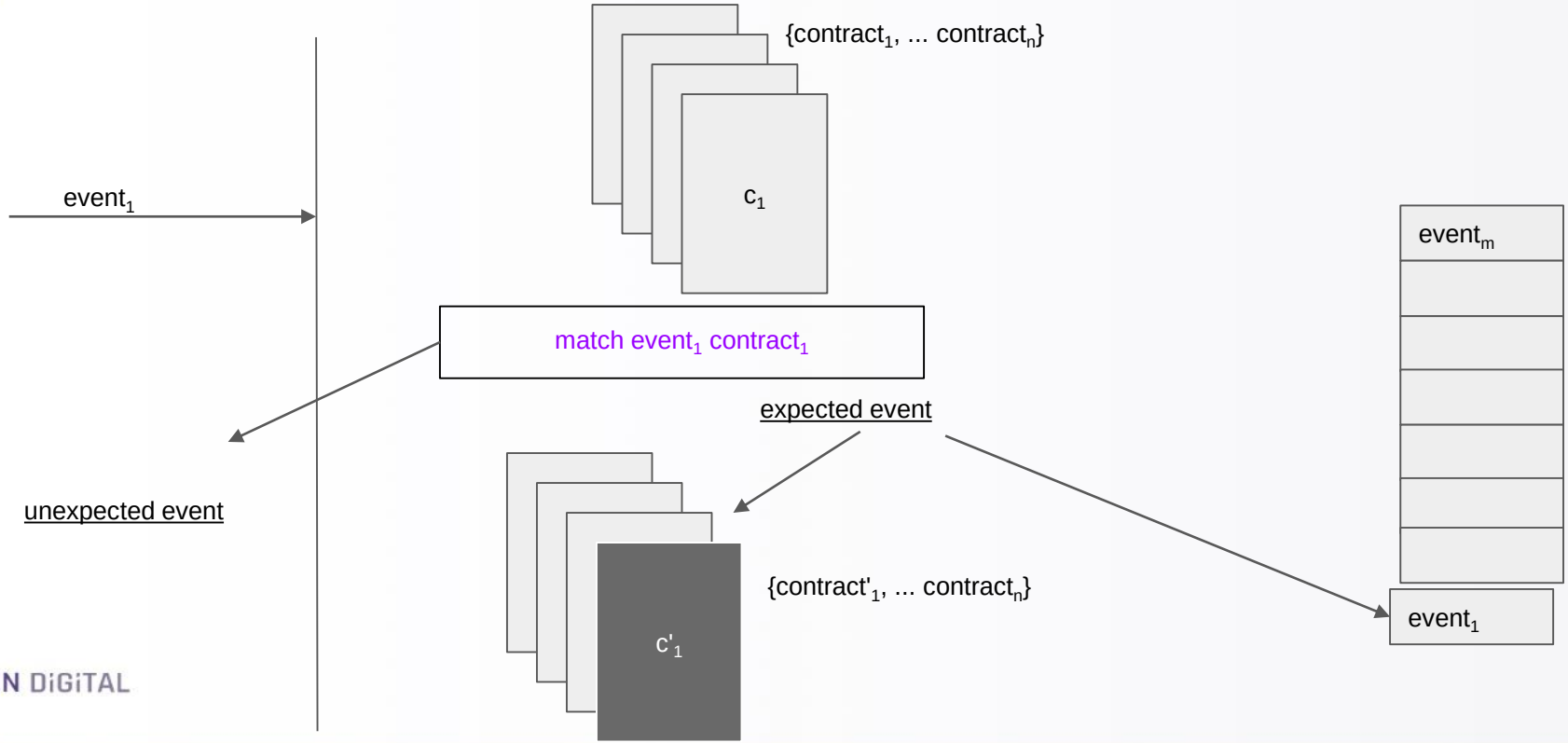






Contract monitoring

- Given contract and event, we can compute the remaining obligations/permissions of occurrence of the action represented by the event (the *residual* contract).
- A contract monitor is a program that accepts events and computes residual contracts (and reports events not expected)



Key observations

- Contracts are analyzable/readable
  - Formal semantics establish a one-one relationship between CSL and mathematical meaning
  - Formal semantics allows (static) verification (conformance to policy/law)
  - Residual contracts are themselves contracts
- Contracts specified using combinators
  - Meaning of contract is compositional. Contracts are spanned through compositions and logic combinators (and, or, not).
  - Libraries of reusable templates/queries/reports can be build for domains of particular interest (mobility, banking, ...)
  - Concrete contracts span possibly large (and/or) wide decision trees
- Contracts are passive (specifications) but are automatable
  - different strategies for fulfilling obligations may be employed by different agents and need not be revealed as part of contract
  - but monitoring execution is possible
  - **and execution is automatable and optimizable!**

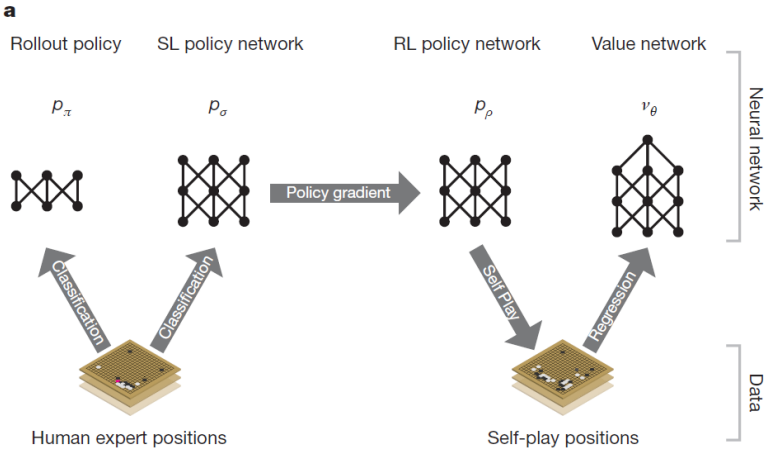
Today AI estimates relationships from past data and predicts best decisions from generalized principles

- **Modern AI is mostly based on deep learning neural networks. Spectacular successes such AlphaGo are based on such technologies.**
- **Neural networks are universal approximators**, i.e. the can approximate any relationship in data (if such relationship exists, danger of overfitting)
- **Neural networks are trained with past data**. For example, the past business flows, contract data, valuations, income, profits, etc. are used to "understand/model/classify" data.
- The **"learned" models** are used to infer **general principles** (approximation of the true behavior)
- The "**learned model"** is then used to **power decision making** from inferred/generalized past behavior.
- The decision model works when the past behavior and future behavior are very similar**. This approach often brings good in average decisions and poor decisions and poor understanding when the situation differs from "average" past**

Excursion: DeepMind and AlphaGo (Google ©)

- ✓ Traditional AI methods, which construct a search tree over all possible positions, don't have a chance in Go. This is because of the sheer number of possible moves and the difficulty of evaluating the strength of each possible board position.
- ✓ AlphaGo therefore combines an advanced tree search with deep neural networks. These neural networks take a description of the Go board as an input and process it through a number of different network layers containing millions of neuron-like connections.
- ✓ One neural network, the "policy network", selects the next move to play. The other neural network, the "value network", predicts the winner of the game.
- ✓ Important observation is that the rules of the game (processes) are completely know. The best move (optimal decision) is the one which improves the value and is selected from the policy network.

[3]



Grand Idea: From formal semantics to AI supported decision support to optimize the best choices in contracts

- ✓ **Deon contract specification language (CSL),** is based on the concept of events (business actions/decisions) by parties which can change path/state of the contract and change value of contracts and accounts. **The CSL specifies the current and future contract/business states**
- ✓ **Deon CSL** possess a **formal semantics**, i.e. one can reason over the meaning of syntax and semantics in mathematical way, i.e. the can **translate the CSL into mathematical equations**
- ✓ **The (abstract syntax) tree,** which spans the algorithmic "walk through" of the execution of contracts and is used to **span the economic decision tree**. The tree span the "rules of the game", the decisions alter the economic value of the firm
- ✓ **The CSL includes a reporting language (RSL)** which allows the user to report over and value contract paths, contract states, and account states
- ✓ The reporting language together with the enumeration of all events, empowers the calculation of the **best economic decision through the principles of dynamic programming** (backward induction). This is only possible for small portfolios of contracts and/or less complex contracts
- ✓ The problem of **finding the best possible** set of **decisions** to maximize the economic value **is the same as playing a difficult board game, e.g. Go or Chess.**
- ✓ ***The network are trained in small situations and approximates the decision function for "large" contracts***

How are the networks trained: On “small” and “few” contract the optimal decision is brute-force computed

Input

CSL (Text)

```
template
LatePayment(paymentDeadline) =
<"Alice"> p: Payment where
p.amount = 100 && p.timestamp <=
paymentDeadline then <"Bob"> d:
Delivery where d.timestamp <
DateTime::addDays p.timestamp 2 &&
.....
```

RSL (text)

```
type Sale: Event { amount:
Float } // Returns "Some e" if
the event is a Sale worth more
than `v`. "None" otherwise val
isSaleOverV = \v -> \(e: Event)
-> type x = e of { Sale -> if
```

Events

[Payment { agent = "Bob", t... = #2018-07-02T13:35:54.567Z#, amount = 100, receiver = "Alice" }]

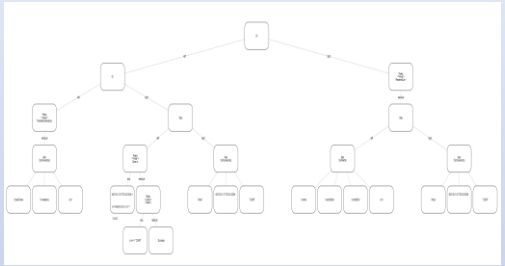
Formal semantics = Mathematical equations (implicit)

Valuation function of decisions

Decision variables

Intermediate State

Tree generator for full trees



Optimal decisions on full trees through dynamic programming since [1]

Decision
f(Contract,Events,Report)

Output

Best optimal next event (business action)

How the decision are made: Input contracts and events, Output is the next best event (action) choice

Input

CSL (Text)

```
template
LatePayment(paymentDeadline) =
<"Alice"> p: Payment where
p.amount = 100 && p.timestamp <=
paymentDeadline then <"Bob"> d:
Delivery where d.timestamp <
DateTime::addDays p.timestamp 2 &&
.....
```

RSL (text)

```
type Sale: Event { amount:
Float } // Returns "Some e" if
the event is a Sale worth more
than `v`. "None" otherwise val
isSaleOverV = \v -> \(e: Event)
-> type x = e of { Sale -> if
```

Events

[Payment { agent = "Bob", timestamp = #2018-07-02T13:35:54.567Z#, amount = 100, receiver = "Alice" }]

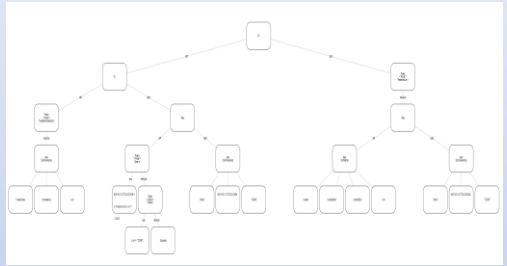
Rules of the "Game"

Values the different decisions

Decisions sets by "players"

Intermediate State

Tree generator (part of RT)



Advanced tree search

Output

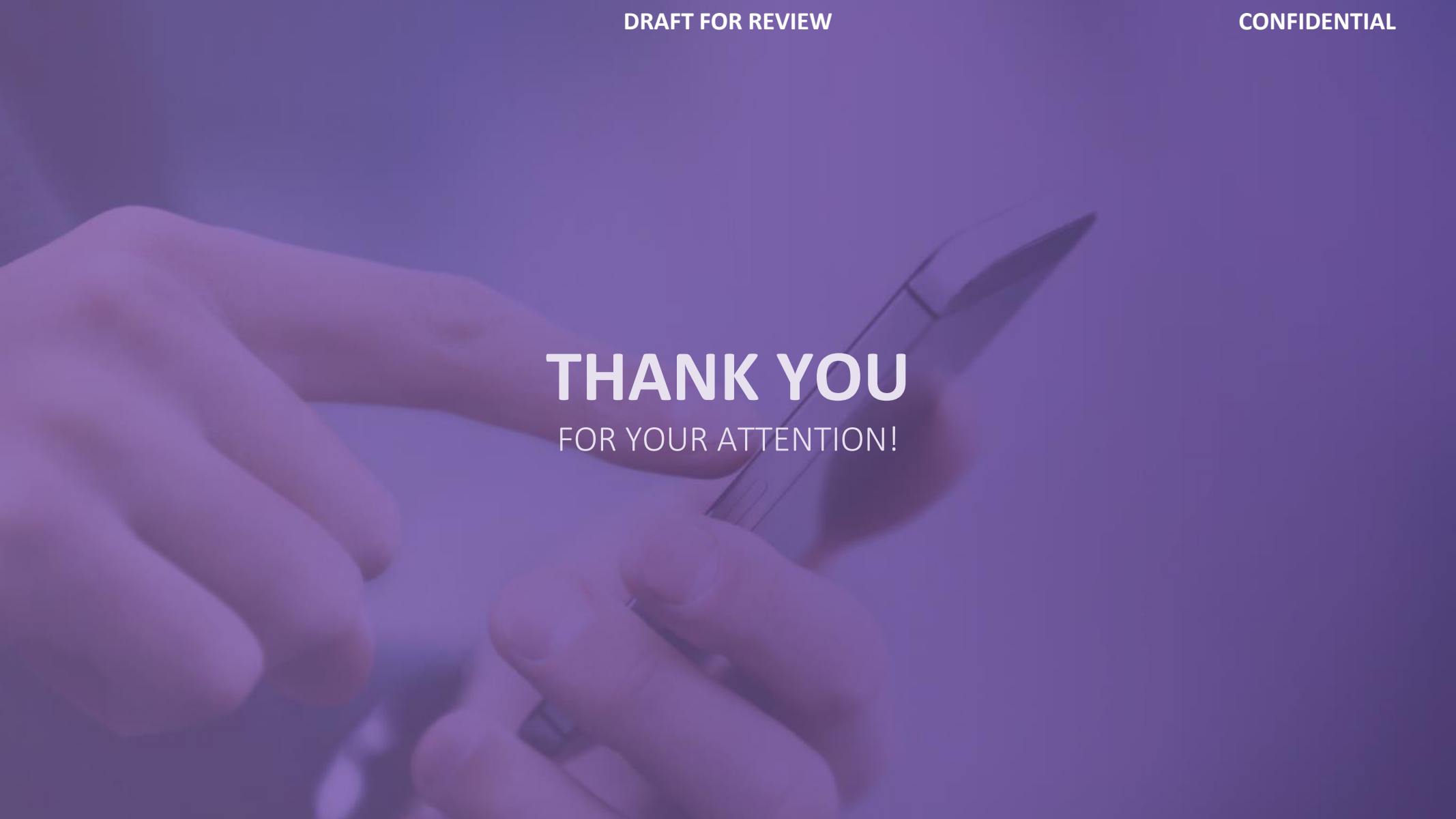
Best optimal next event (business action)

Comparison to AlphaGo and Deon Digital Decisions Analytics

- ✓ **The contracts and events = Rules of the Game and Moves**
- ✓ **The reporting / valuation = Goal, i.e. Winning / good Positions**
- ✓ **Past Games / Random Games = Optimal decision on “small” contracts**
- ✓ **Current board position = Set of contracts, accounts value, events**
- ✓ **Next best move = Best next event or best next contract**

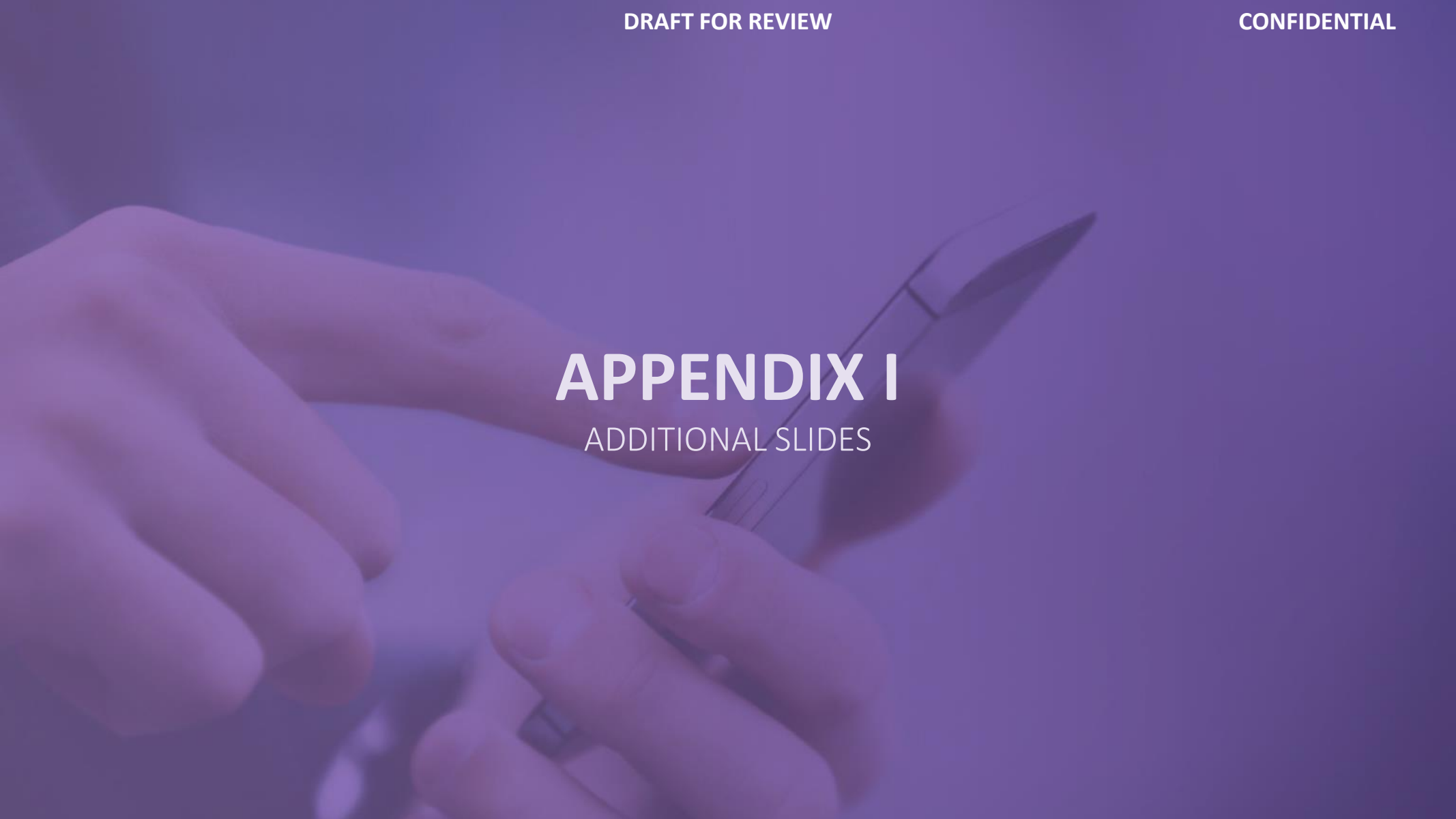
Winning features of Deon Digital development suite: Mathematical analysis and decision making on contracts

- ✓ **Human and computer readable contract language** (Ricardian contracts) through an easy to understand and implement contract language
- ✓ **Full reporting language** to report over past event, value contracts and build arbitrarily complex state machines
- ✓ **Higher security** through separation of contract design and contract execution
- ✓ **Higher safety** through specialist language for contracts instead of Turing complete language
- ✓ **Avoidance of design failures, bugs, and errors** through **formal semantics** and **formal verification**, i.e. features of contracts can be mathematically proven
- ✓ The mathematical analysis of our programs **empowers full future looking decision analysis** on the actual portfolio of contracts, past events and value of accounts
- ✓ **Easy design of complex processes** through the interaction of contract and reporting language and the notion of events (business actions)
- ✓ **Advanced analytics** gives a valuation to contract future decisions and span the tree of future contract states
- ✓ **Advanced decision making** using **deep neural network approximates the** decision function but relies on the “actual” future business states and NOT on some past behavior



THANK YOU

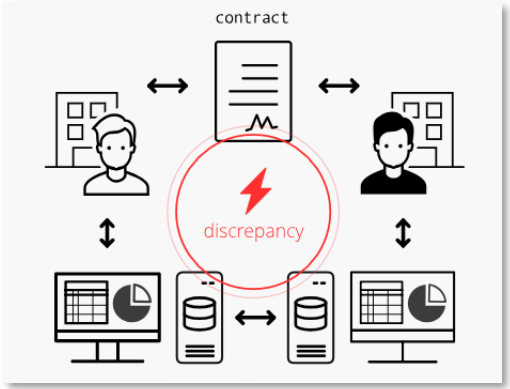
FOR YOUR ATTENTION!



APPENDIX I

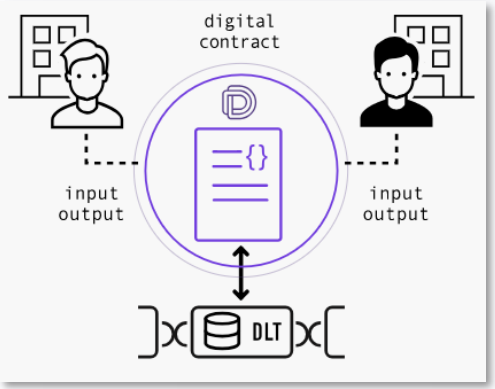
ADDITIONAL SLIDES

Build ‘intelligent’ systems, that fully digitalizes rights, processes, and obligations



- **Contracts on paper**, IT systems cannot read them
- Multiple IT systems execute **processes separately from the contracts**

May lead to…
- data **redundancies** and **inconsistencies**,
- increased **reconciliation effort**, **inefficiencies**,
- **high number of interfaces** and **increased complexity**
- **slow reaction time** to new business opportunities



- **Contracts/processes are fully <u>digitalized</u>**; man and machine can read them
- **One single source of truth**; digital contracts contain the obligations, rights and prohibitions of participants, clearly stating **who does what under which conditions**
- Distributed ledger technology ensures **transparency**, **surefire transaction history** and **instant execution**

Digitizing contracts

Normal contract:

- piece of paper (pdf) describing future allowed and required actions of participants in contract

(Digital/Deon) contract:

- "readable" *specification* of future allowed and required actions of participants in contract

Smart contract:

- code that *runs* and is given mandate to perform future actions on behalf of participants in contract

Deon Digital product suite today

Programs

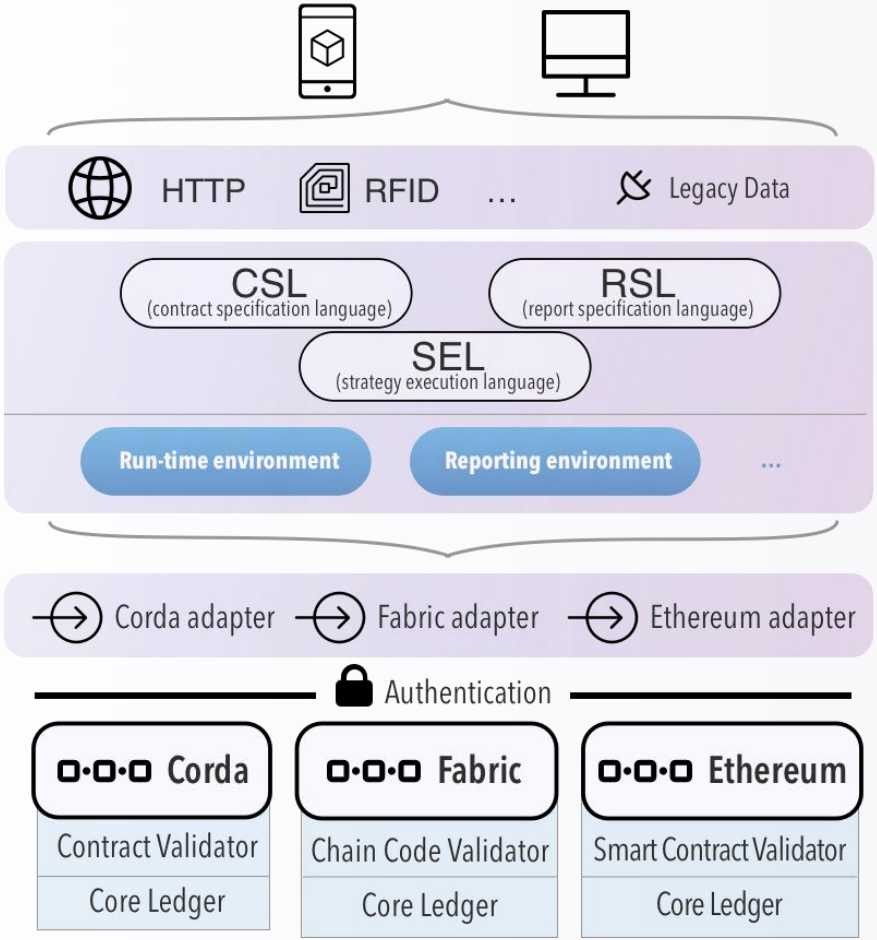
External interfaces

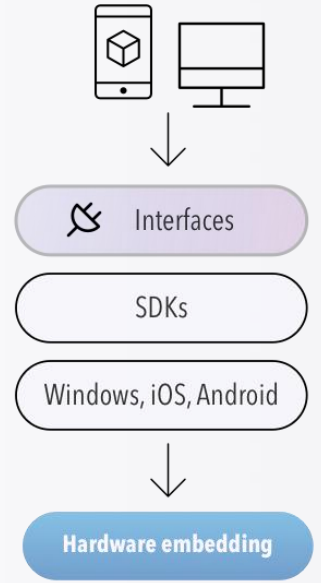
Software language

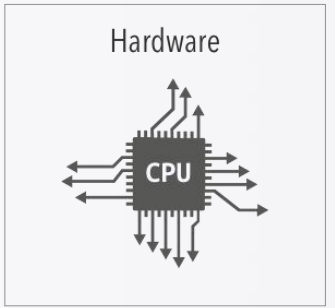
Operating System

Ledger / Hardware Connectors

Ledger Technology as Hardware

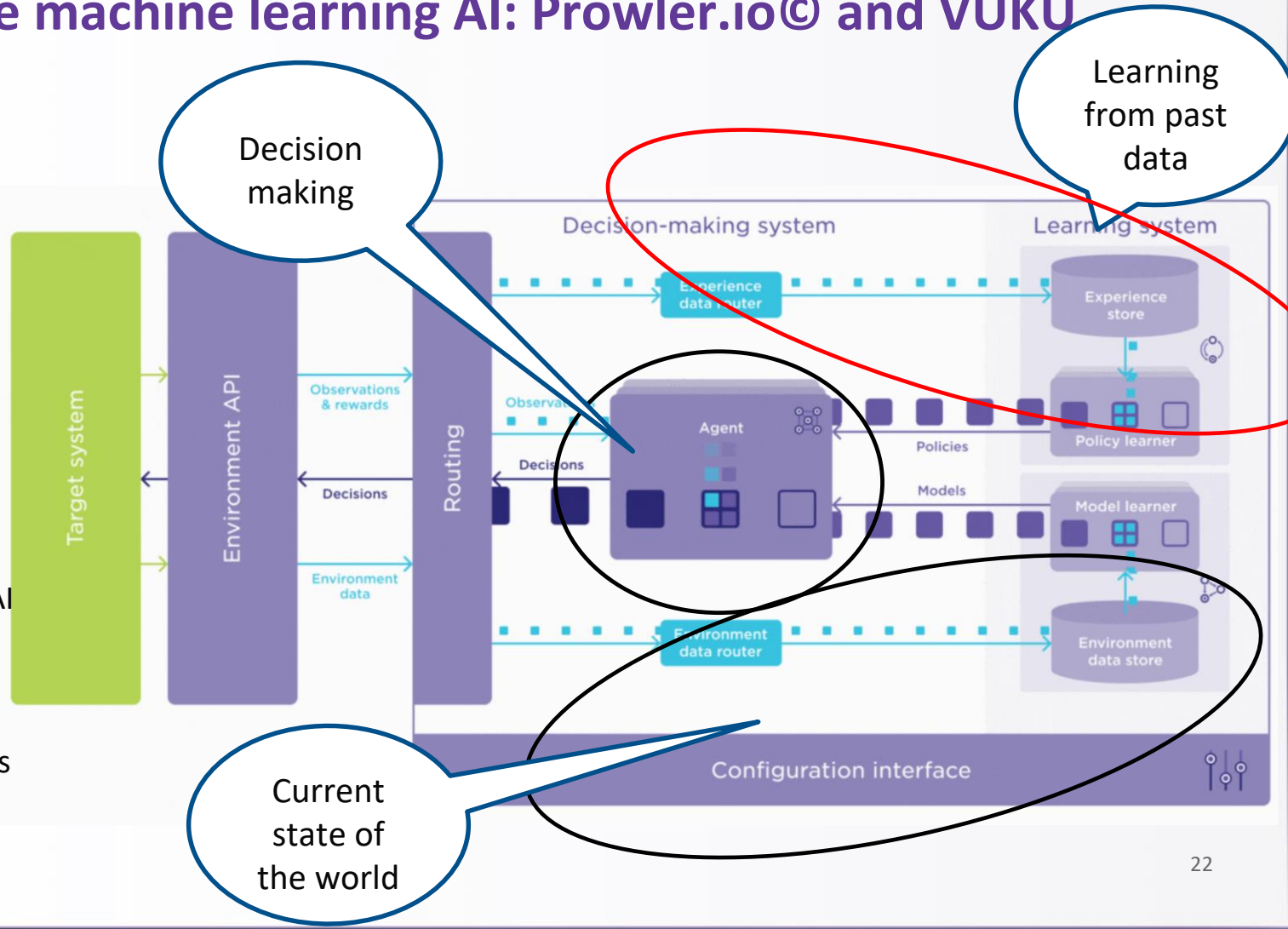




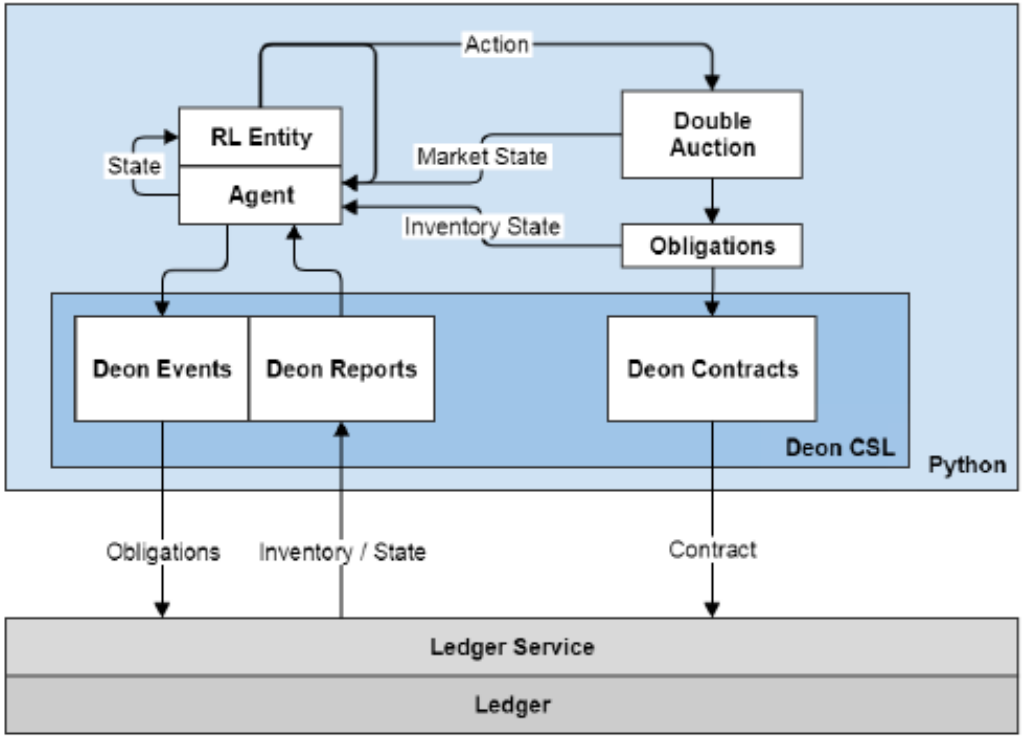


Example machine learning AI: Prowler.io© and VUKU

- VUKU™ is ideally suited for autonomous decision-making in complex, dynamic and uncertain environment.
- It includes a comprehensive set of patent-pending AI tools and technologies to address varied requirements across all these industries.



Example: Small-scale Economy Simulation on a distributed Ledger [2]



TEAM
MANAGEMENT

Copenhagen

Prof. Dr. Fritz Henglein
Head of R&D

>25 years research experience in algo-rithmic, semantic and logical aspects of programming languages. Prof. for Programming Lang. at University of Copenhagen and NYU. PhD in Comp. Science.

Oliver Bussmann
Advisory Board Member

>25 years experience as global thought leader in fintech. Held influential roles at UBS, SAP, Allianz, Deutsche Bank. CIO at IBM managing large transformations.

Dirk Sebald
CEO & Founder

>30 years experience in technology and financial markets - CEO & founder of axzoom - Group Head Innovation SIX Group, CEO of Adesso (IPO) - CEO of Cognos (acquired by IBM) - MBA.

Munich

Pascal Forster
Advisory Board

>20 years of experience in recruiting CEOs, board members, and senior execs. Multiple Cofounder (e.g., World. Minds & XU Expo-nential University). Angel-investor with active involvement in multiple startups.

ZURICH

Tarek Selim
CFO

>20 years international investment & banking. Co-founder Arabesque. Experience in complex financial transactions for UBS Investment Bank, Goldman Sachs and Lehman Brothers.

Dr. Florian Herzog
CTO & Founder

>15 year work experience in financial markets, Co-Founder of swissQuant Group and CEO. Research at ETH Zurich. PhD in Mathematical Finance from ETH Zurich.

Alex Seidel
Board Member & Head of HR

>35 years experience in consumer goods CEO & Chairman of Unilever Switzerland, board memberships across indutries. Fintech investor with stakes in multiple startups.

Literature

[1] Jiequn Han1 and Weinan “Deep Learning Approximation for Stochastic Control Problems”, arXiv:1611.07422v1

[2] D. Hänggi “Small-scale Economy Simulation on a distributed Ledger”, Master Thesis ETH Zurich, 2018

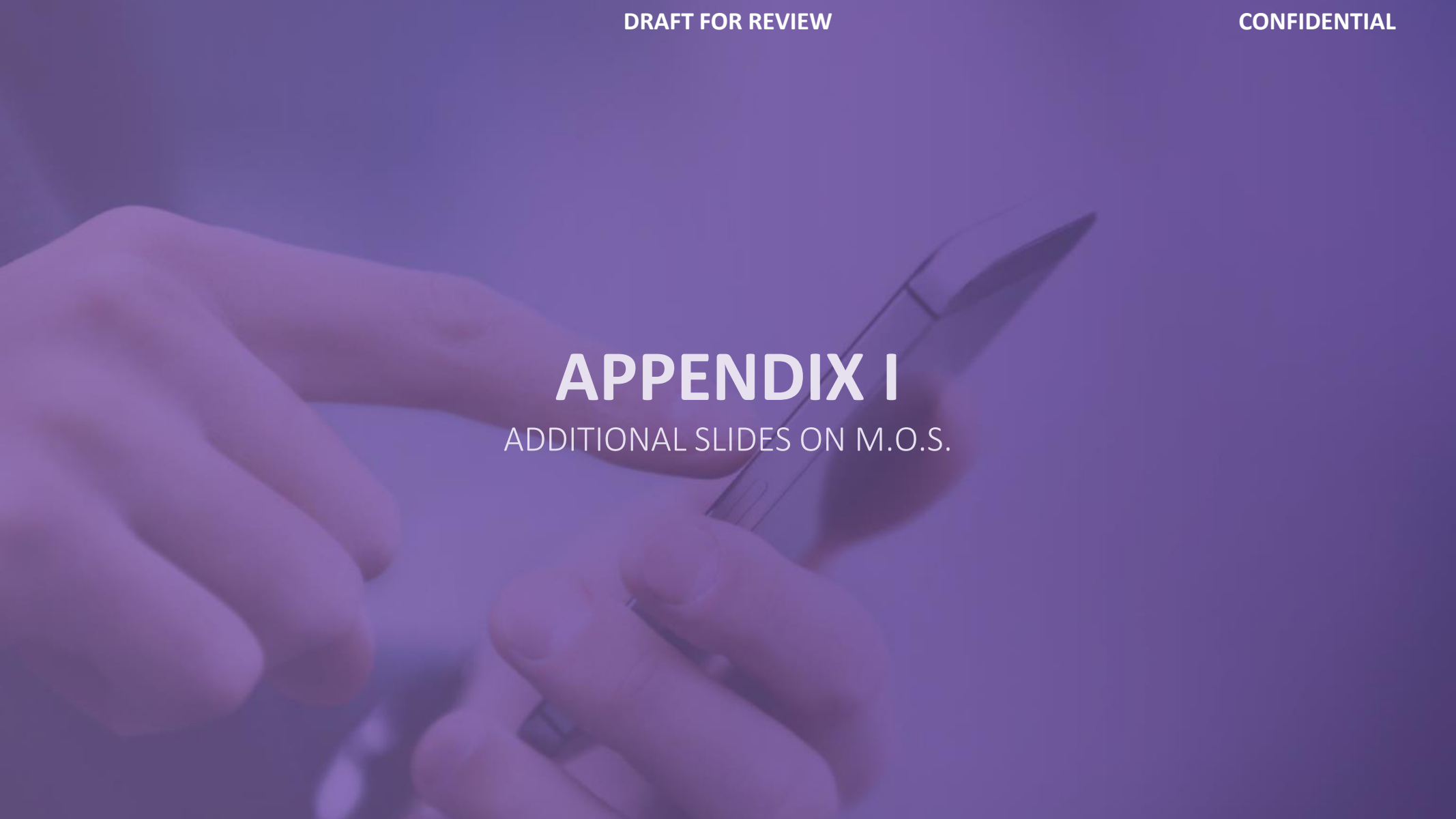
[3] David Silber et al. “Mastering the game of Go with deep neural networks and tree search”, Nature, p 484 -504, NATURE, VOL 529, 28 JANUARY 2016

The powerful “Automated Robotic Coding”

executes several technical steps in the background

- Digital objects (i.e., ontology) Representing all parties and contracts
- Data model to describe the Properties of each object
- Network including topology (i.e., network structure) to set up the distributed ledger
- Distributed ledger nodes, notaries, network manager, transaction ordering service, etc.
- Database and access rights to record all events in all nodes
- Processing business logic in the network to program distributed ledger protocols
- UI connectivity via REST interfaces
- Web service including development user interface (UI) to access and use ledger

"Deon’s technology covers business logic and a fully operational backend that can be set up within only a few lines of code - all that is required on top is user-friendly client interfaces.”

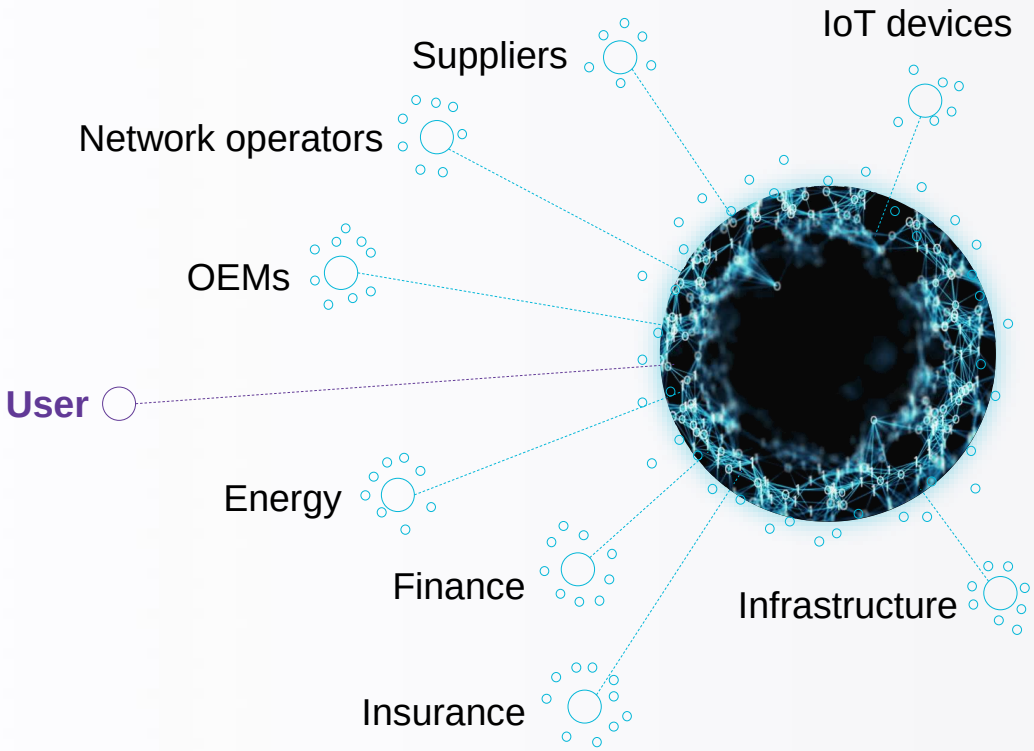


APPENDIX I

ADDITIONAL SLIDES ON M.O.S.

Today, players of the domain form scattered nodes
in the Mobility Universe

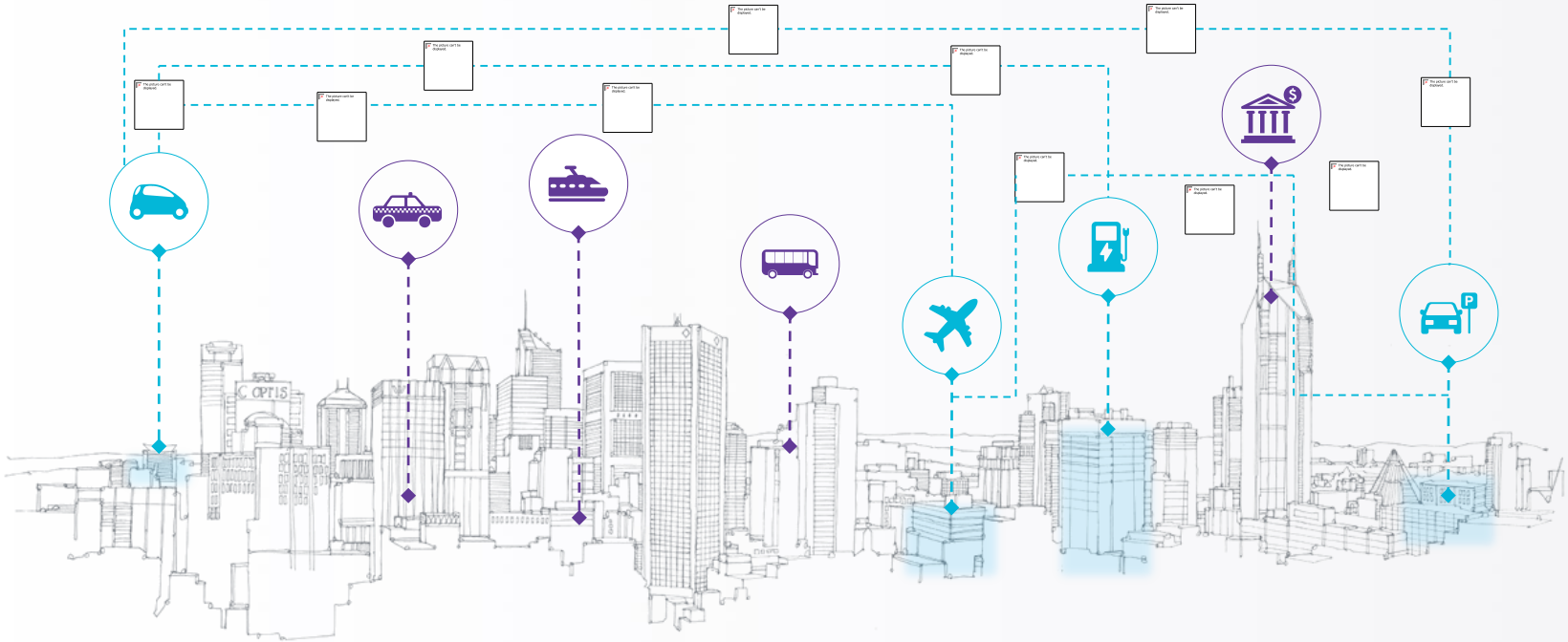
Mobility Domain exists for 100s of years. It is getting more **fragmented**, each business field **expanding** and drifts apart into **silos**



Source: Daimler

Complex network of business nodes for future mobility

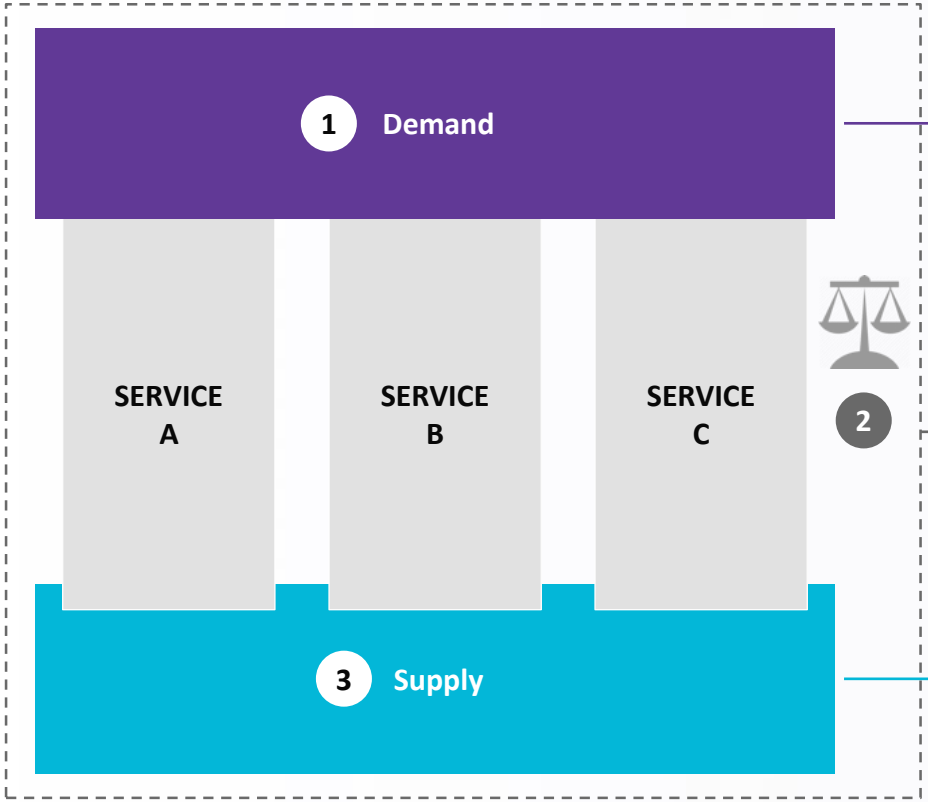
Collaboration of services exchanging contracts and data



Source: Daimler

We form a standard by aggregating supply

in a collaborative approach



1 Aggregate demand (e.g., UBER)

Leads to dependencies and expensive contributions to a proprietary standard (e.g., Google, Airbnb, etc.)

2 Aggregate via regulation (e.g., SEPA)

Authority-induced standards lead to expensive adaptions in the domains e.g., SEPA

3 Aggregate supply/Collaborate (e.g., ISO)

Flexible enabler technology generates efficiencies and agility for businesses with low entry barriers; business expansion

Source: Daimler

Any viable Mobility Standard requires…

an open, collaborative approach to overcome today‘s constraints and incorporate existing legacy

Today’s Challenges

- Huge size and number of players
- Legacy and Monopolies
- IoT interaction with Hardware

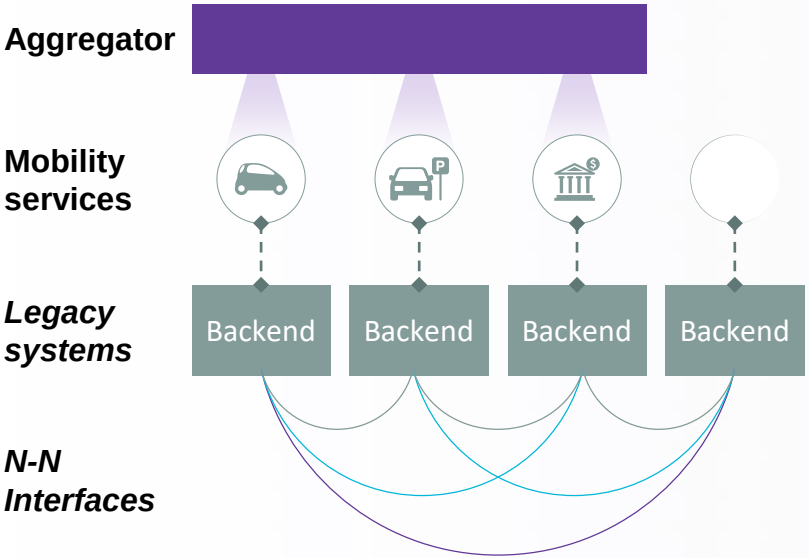
Future Requirements

- Mutual tech basis and standards
- Automation and operational efficiency
- Maximum autonomy for existing businesses
- Enabling collaboration
- Full integration of mobility devices
- Scalability and flexibility

Source: Daimler

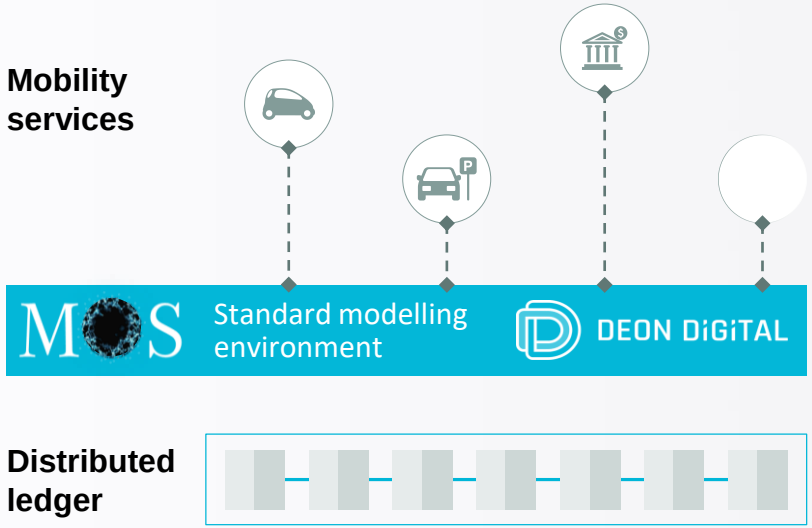
Mobility operating system makes the difference

Today's challenges



Collaboration takes place by sending individual "requests" through the service APIs of all participants

Future collaboration



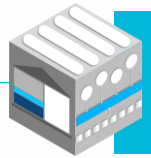
- ✓ **Standardized collaboration by reusing contracts and objects**
- ✓ **Logging in distribution ledger privacy and trust level**

Source: Daimler

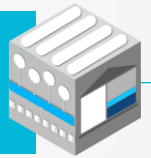
The vision of the MOS collaborative initiative

Our vision for the **global mobility & transportation industry** is a **collaborative** and **discrimination-free digital market infrastructure** with **common standards** for **all players:**

a **Mobility Operating System**



Call for interested partners to join MOS collaborative initiative



Source: Daimler

TECH
SOLUTIONS

Contract Specification Language (CSL)	• Process-oriented language to design and implement contracts and business processes • Easy to use and fast to adapt with minimal coding know-how • Defines contracts (therefore processes) by a finite state model • A contract is passive and does not execute itself, it merely reacts to events (by accepting or rejecting them)
Report Specification Language (RSL)	• The user can define reports (i.e. queries) that can be used within contracts or serve as standalone reports • Reports query past events and contract data
Strategy Execution Language (SEL)	• A user can build automated components based on a combination of reports and workflows/contracts • SEL "talks to the outside world" and runs at the individual node level • For example, an executed payment can lead to an automated delivery, according to the corresponding workflow
Operating System and run-time environment	• Interprets and executes the CSL, RSL and SEL codes (i.e. robotically builds applications) • Connects the node to the ledger (via the adapter) and to the UI/UX by an autocoded API • Serves as an interface between the contracts/ledger and classical IT systems (databases) • An event processing engine which accepts only specified events and rejects all other • events (errors, frauds, etc.)
Blockchain Adapters	• Translates the contract and reporting logic to blockchain technology • Makes the contract and process specifications independent of the blockchain implementation, i.e. the technology is ledger agnostic • The contracts can be implemented in different DLT technologies