Smart City Digital Twins: A Survey of Key Technologies

Eero Immonen*, Tero Villman, Michael Lindholm***, Jari Kaivo-oja****

*Turku University of Applied Sciences, Finland

** University of Turku, Finland

*** Turku Science Park Ltd, Finland

Contents

Smart City Digital Twins
History
Standardization
Survey results
Future outlook
Conclusions

TURKU AMK

Smart City Digital Twins (SCDT)

TURKU BUSINESS REGION

TURKU AMK

- **Smart City**
  - *"technologically modern urban area that uses different types of electronic methods and sensors to collect specific data."* – Wikipedia
    -> This is done **for the benefit of residents**: better decision-making, enhanced efficiency, sustainability, quality of life …

- **Digital Twin**
  - *computer replications of physical devices, systems or processes connected both ways to the physical domain in real time*
    -> Key enabling technology concept for Smart Cities; integral part of the *Industry 4.0* paradigm

- **Smart City Digital Twins (SCDT)** facilitate **prediction of complex scenarios with multi-user interactions**, based on real-time data and numerical analytics, **before they take place in the real physical world**.

History highlights

- 1980s
  - Personal computers, analogue point-to-point communications, industrial automation systems
- 1990s
  - Internet, e-mail, digital distributed communication; city information first available online in **manually updated** announcements (alongside printed publications).
- **2000s**
  - Industrial HiL, cloud computing (e.g. AWS 2006), onset of massively interactive client-server software (e.g. Facebook); towards **real-time city information**, e.g. in public transport tables. First API's starting to appear.
- **2010s**
  - Digital simulation platforms, IoT, mobile internet and smart phones; **2-way access to real-time city information anywhere**, including elementary information synthesis (e.g. server back-end gathering data from several sources)
- **2020s**
  - Artificial intelligence and complex simulation/prediction models; **vastly interconnected** city information systems or capable of acquiring distributed real-time data, analysis and even **synthesis for decision-making**

Standardization for SCDTs

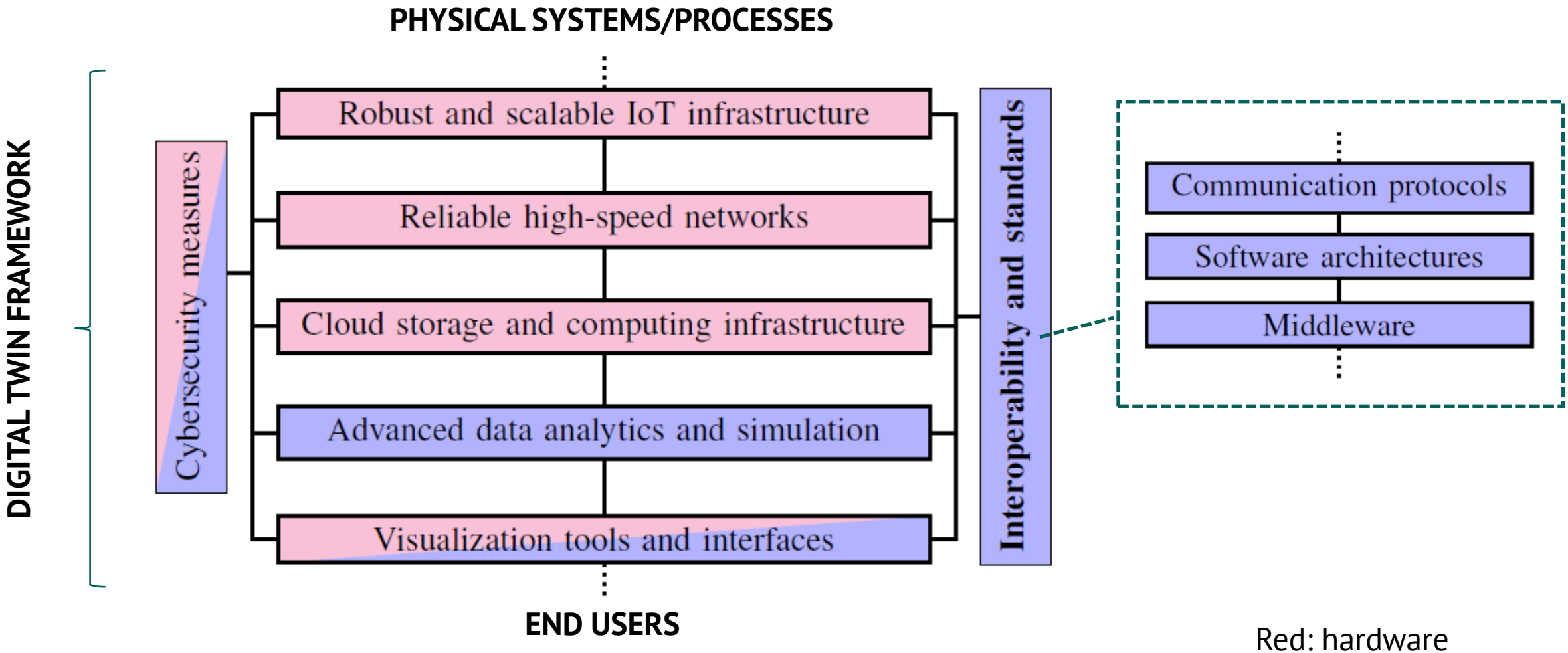
- Regardless of application, **standardization always aims at "not reinventing the wheel"**

- For SCDTs, standardization refers to **modular** software and hardware solutions that are
  - Reusable
  - Interoperable
  - Low-complexity (relatively speaking)
  - Can be rapidly and collaboratively developed in isolation of the end application
  - "Plug-and-play"

- For SCDTs, standardization **benefits**
  - Flexibility – use any suitable module,
  - Maintainability – modifications and upgrades at module level
  - Scalability

–> **Managing the growth of smart city information systems!**

Taxonomy of SCDT standardization



Red: hardware
Blue: software

Standardized communication

TURKU BUSINESS REGION

TURKU AMK

TABLE I: Comparison of communication protocols for SCDTs.

Protocol	Features	Advantages	Disadvantages
MQTT (Message Queuing Telemetry Transport)	Publish-subscribe model for real-time data communication over TCP. Lightweight design.	Real-time communication, scalability, efficient data distribution in centralized applications.	Overheads may affect large-scale/high-frequency applications, only 3 levels of Quality-of-Service (QoS).
CoAP (Constrained Application Protocol)	Request-response model for resource-constrained devices. Focus on efficiency for limited processing power and bandwidth.	Fast UDP communication, lightweight, resource-constrained devices support.	Limited adoption and support in existing middleware, reliability of UDP (vs. TCP), potential issues in handling large data payloads.
LWM2M (Lightweight Machine-to-Machine)	Device management capabilities, standardized registration, monitoring, firmware updates.	Standardization of IoT device management.	Learning curve, built on top of CoAP (see its disadvantages above)
DDS (Data Distribution Service)	Real-time peer-to-peer data exchange, encryption.	Efficient communication in distributed (decentralized) applications, control over QoS.	Complexity, overheads, best suited for large-scale applications
WebSockets	Persistent full-duplex point-to-point communication over TCP. Real-time updates for user interfaces.	Real-time interactions, user interface updates, support for large messages.	Best suited for real-time dashboards and user interactions, requires a persistent connection (resources).

Standardized architectures

TABLE II: Comparison of software architectures for SCDT

Architecture	Features	Advantages	Disadvantages
Microservices Architecture	Decomposes application into small, deployable services.	Enhances agility, scalability, and ease of maintenance by separate technology stacks.	Requires careful management of inter-service communication and potential performance overhead.
Event-Driven Architecture	Enables asynchronous communication, real-time responses, and component decoupling through events.	Supports real-time interactions, scalability, and flexibility in system components.	Complex event handling, potential increased system complexity, and learning curve for event-driven programming.
Edge Computing Architecture	Processes data at the network edge, reducing latency and enabling real-time decision-making.	Low latency, reduced data transmission, improved responsiveness.	Limited processing power, potential security challenges at the edge, requires managing distributed resources.
Hybrid Cloud Architecture	Combines on-premises and cloud resources for flexibility in optimization.	Scalability, cost-efficiency, data residency compliance.	Requires integration between on-premises and cloud systems, potential data synchronization challenges.
Data Lake Architecture	Provides a centralized repository for storing and analyzing large volumes of raw data.	Supports data-intensive applications, data exploration, and analytics.	Data governance challenges, potential data silos, requires proper data management strategies.

Standardized middleware: co-simulation

TABLE III: Comparison of open-source co-simulation middleware solutions for Smart City Digital Twins

Solution	Features	Advantages	Disadvantages
ModelConductor eXtend	FMI support, real-time data collection, asynchronous processing, queues	Multi-fidelity simulation models as digital twins, asynchronous data streams	Only tested for traditional engineering applications
OMSimulator	Support for Modelica language, FMI support, ordinary and delayed (transmission line) connections	Versatile model library (OpenModelica) also supporting numerical optimization, active user community	Learning curve, system integration challenging (or expensive)
DCP	Discrete-state machine, real-time and non-real-time operation	Parallel and distributed computing, large-scale simulations	Lack of widespread compatible simulation software

Standardized middleware: generic solutions

TURKU BUSINESS REGION

TURKU AMK

TABLE IV: Comparison of generic open-source middleware solutions for Smart City Digital Twins

Solution	Features	Advantages	Disadvantages
Eclipse OM2M	Device connectivity, data exchange, event processing, support for oneM2M standard	Modular and standardized, real-time interaction with physical systems	Steep learning curve for beginners
FIWARE	Standardized APIs, FIWARE Context Broker, NGSI APIs	Extensive ecosystem, modular and scalable, real-time data management	Requires additional configuration for specific use cases
Eclipse IoT	Eclipse Hono (device connectivity), Eclipse Kura (edge computing), Eclipse Kapua (IoT data management)	Community-driven, comprehensive IoT framework, real-time integration with physical systems	Requires familiarity with Eclipse ecosystem
ThingsBoard	Device management, data collection, real-time visualization	User-friendly interface, rule-based event processing, real-time connectivity	Limited scalability for large-scale deployments

TURKU AMK

So far so good

BUT.....

We need to think about

- Strategies
- Business models
- Organizational structures
- Processes
- Skills

When aiming for larger SCDT’s

What about software standards?

TURKU BUSINESS REGION

TURKU AMK

- **Software standardization are achievable with underlaying data structure standards**

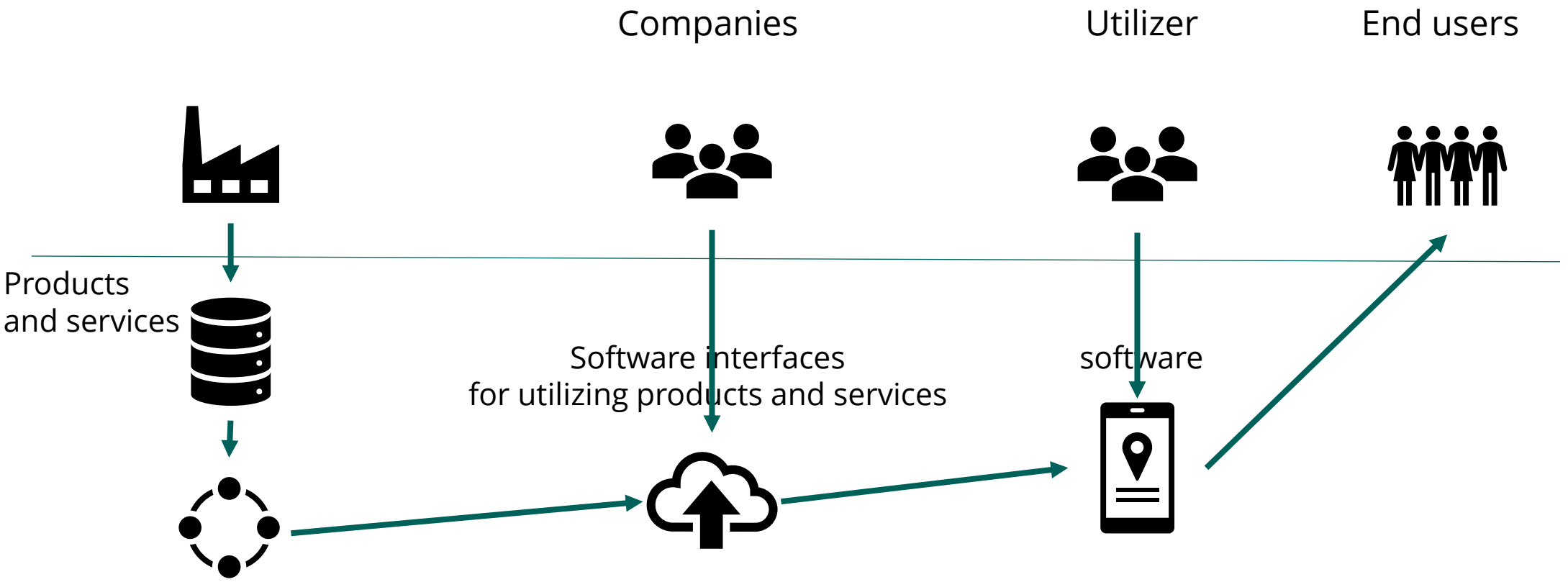
- **High level standards exist but these do not**
  - Define software or algorithms
  - Nor the data structures for business models

- **Defining the functionality and building the software in "blocks" can resolve much e.g.**
  - Composable software
  - API (Application Program Interface) based solutions

  => We need research, planning and cooperation

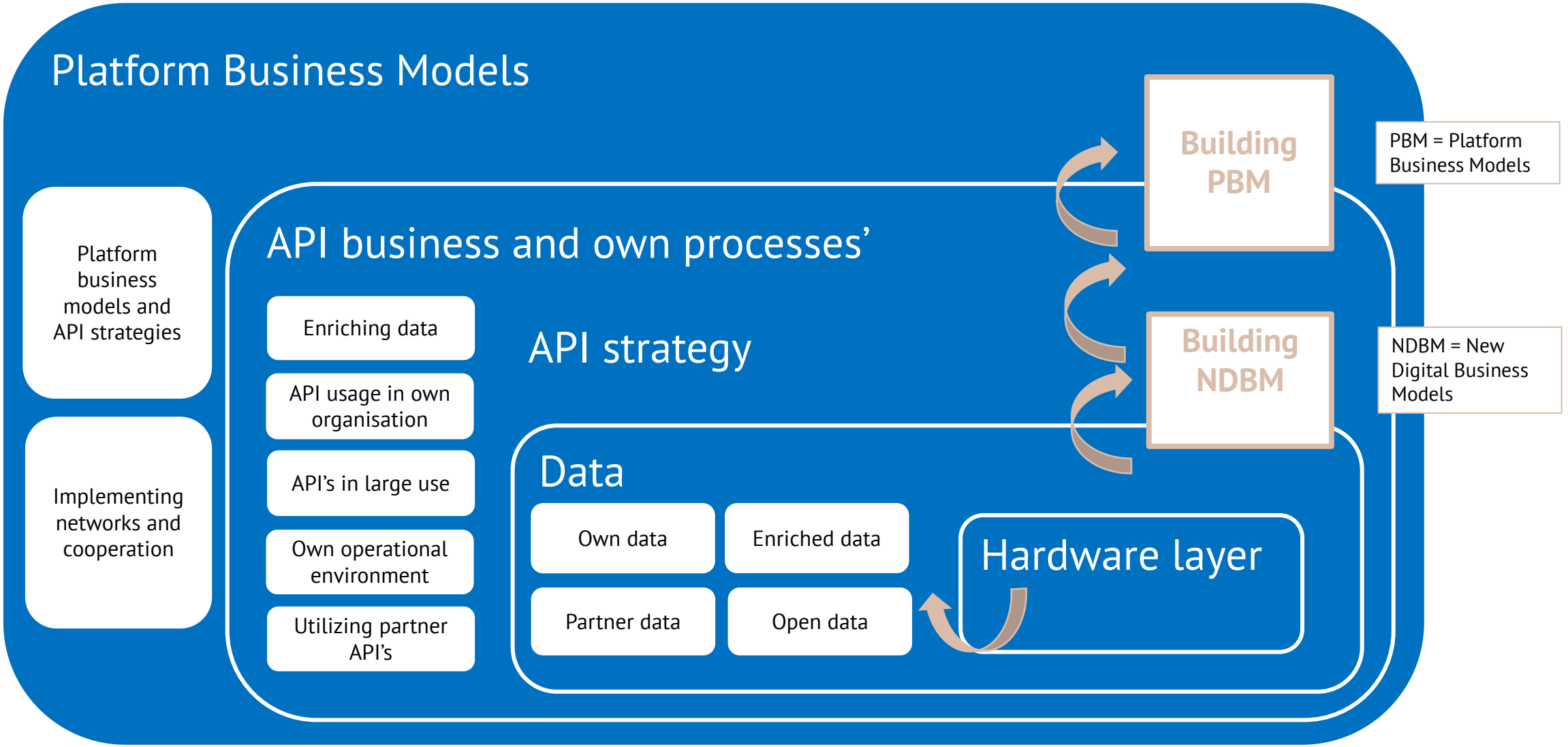
Composable design, or composable architecture, is an efficient way to create software systems composed of freely interchangeable components. These modules work together seamlessly to form a unified whole system. With this approach, developers can build more robust and reliable applications quickly and efficiently.

API principles

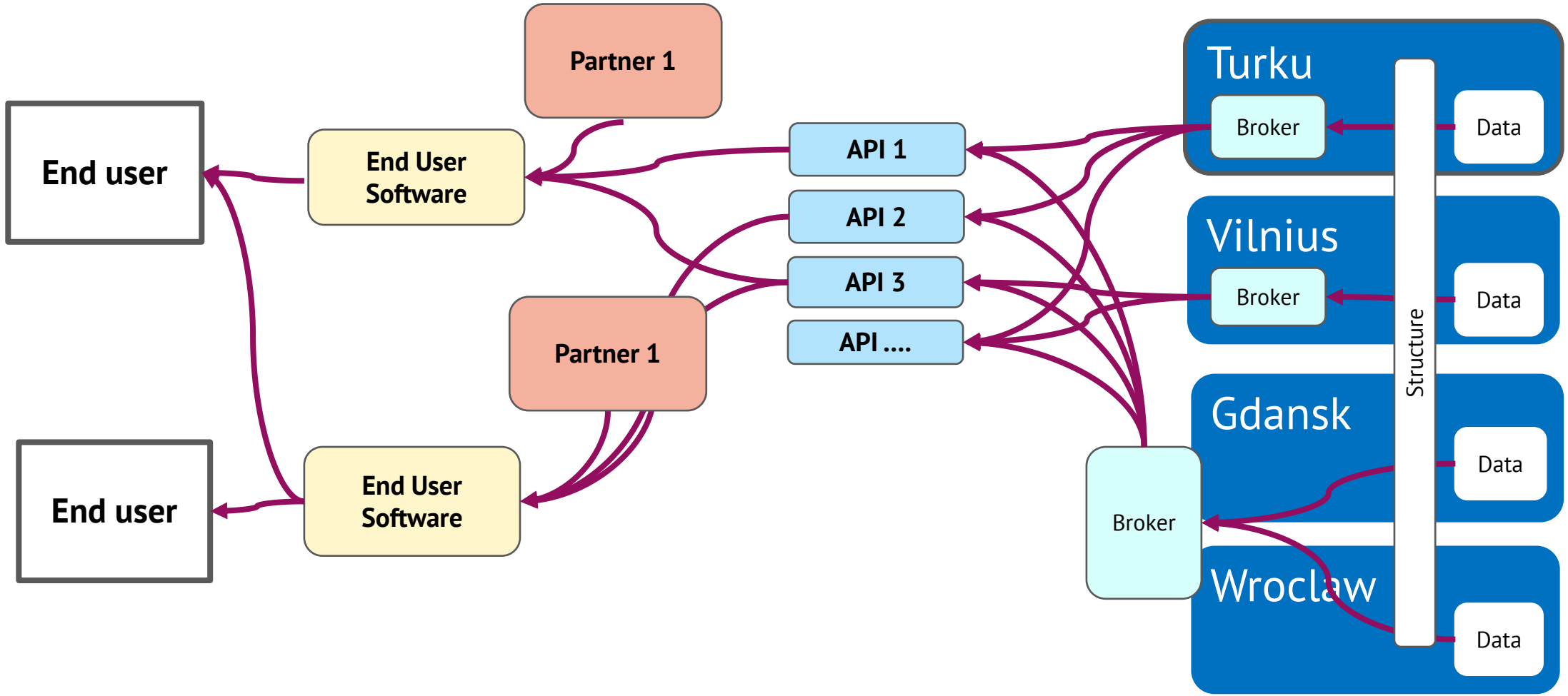


[The Finnish public sector API strategy 2022]

Build business models with API's



An example of a city model with API's



Conclusions

- **Standardization is the key to a certain extent for ensuring compatibility between solutions, we also need cooperation, strong strategies, organizations and skills**

- **We do need to agree on "standardizing" mutual city data structures**
  - We have similar needs but different data sources and different systems
  - Find out joint API strategies and test these in the near future
  - Working towards the visions which is what we are researching now
  - Remember Strategies, Business models, Organizational structures, Processes, Skills)

- **Benefits of working towards mutual goals are obvious**
  - We can agree on standardized API based development as well as goals
  - Structured plans and development have large possibilities and will add value for our regions

<u>We can and will develop Smart City Digital Twins together!</u>

Thank you!
Kiitos!

TURKU AMK

Michael Lindholm
Michael.lindholm@turkubusinessregion.com
+358 40 502 0089

TURKU BUSINESS REGION