Architecture Transition To ”The Ball”

Kalle Launiala / ProtonIT Oy
”The Ball” / Caloom Models

kalle.launiala@protonit.net
+358445575665

Overview of contents

- 1. Technical/Logical Architecture of Single Customer Solution
  - Technical Positioning of Expertise
  - Network of providers enables technical expertise to focus
    - For a superior complete solution for end customer

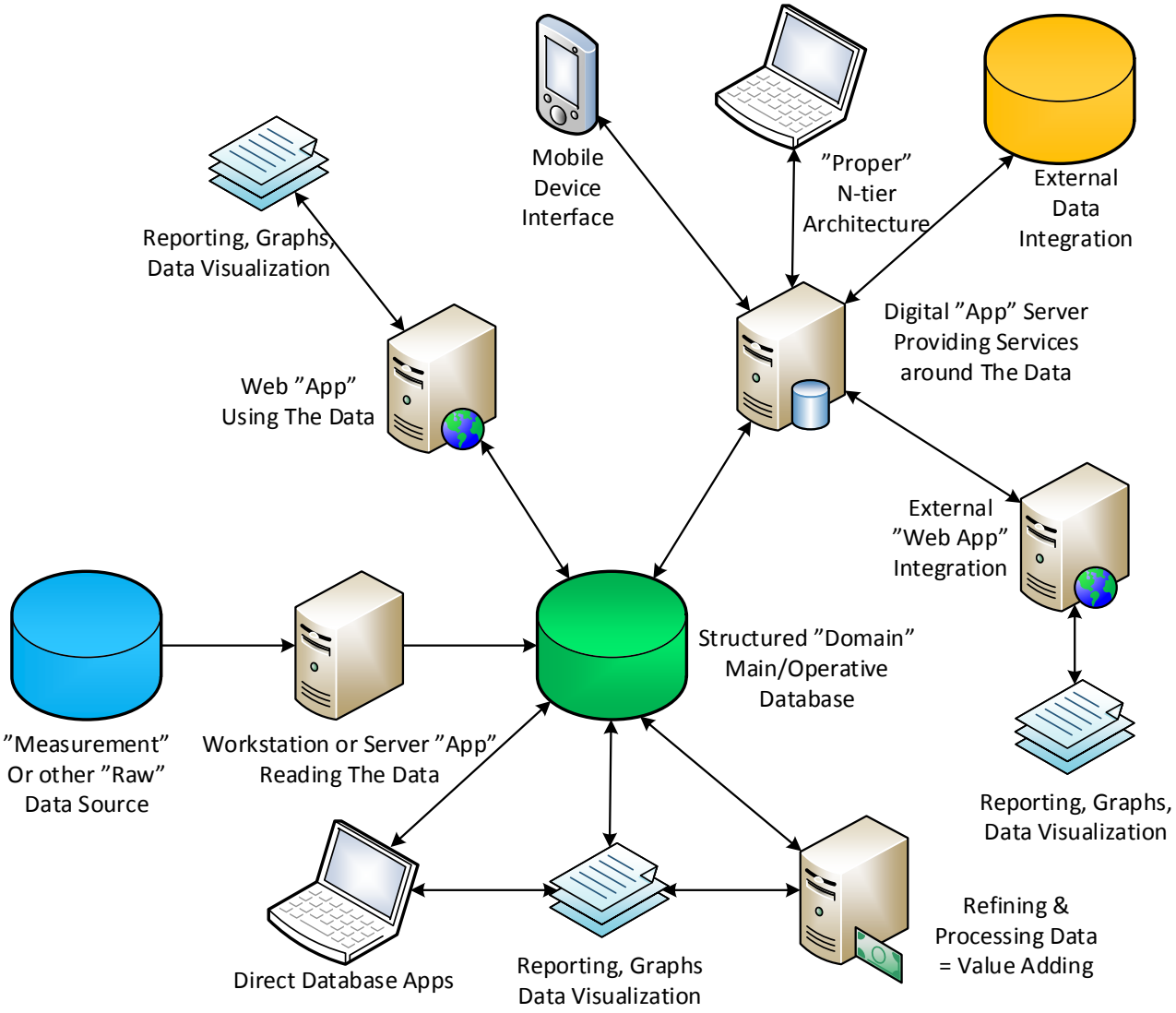
- 2. User/Customer/Consumer/Citizen Perspective
  - Scenario from City's services towards its citizens and companies
  - Siloed system-centered perspective
  - Change
  - Means & Benefits

- 3. Migration from existing systems towards "The Ball"
  - Including the digital service network

Technical

Positioning of Technical Expertise

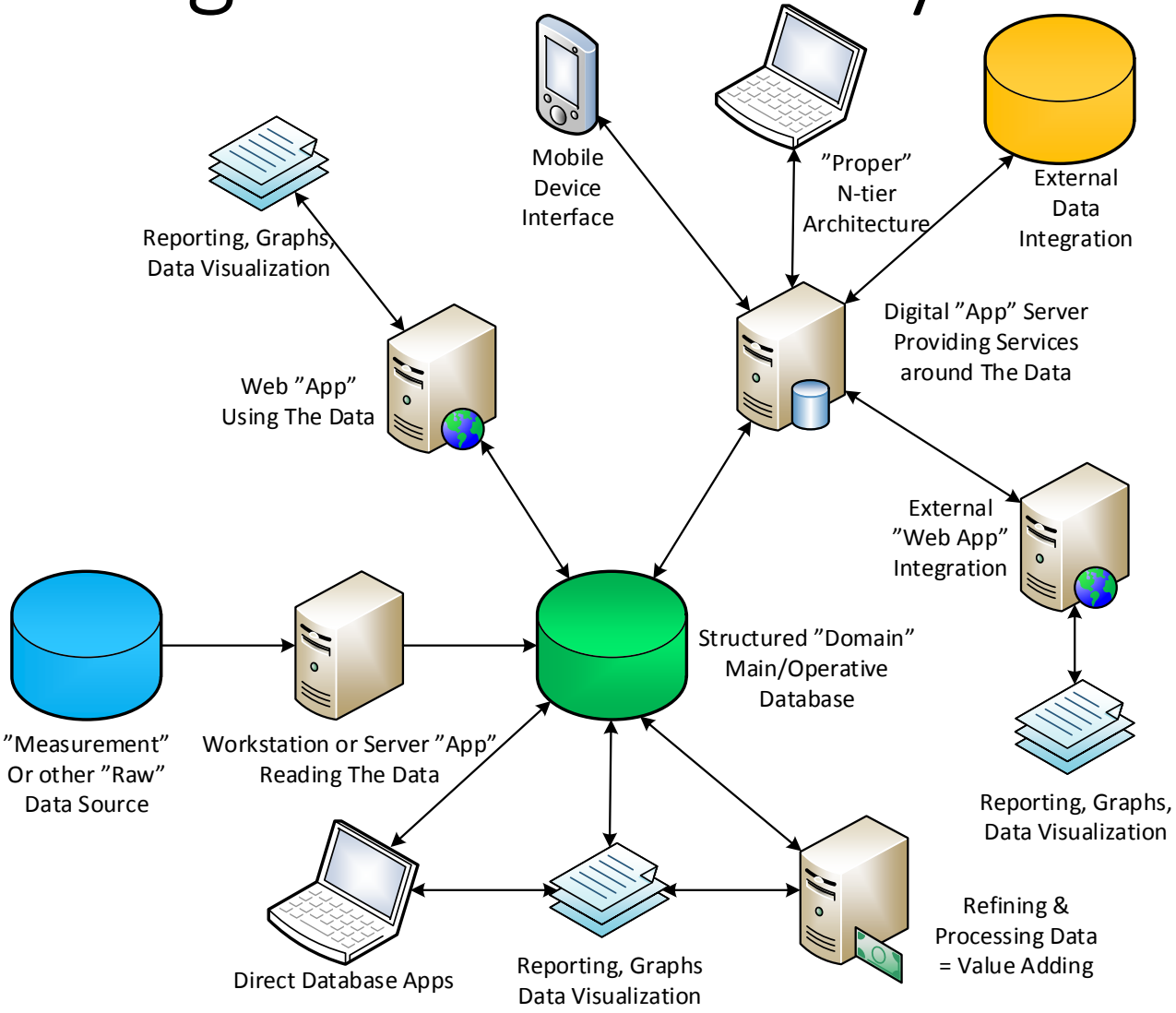
Traditional Technical Architecture



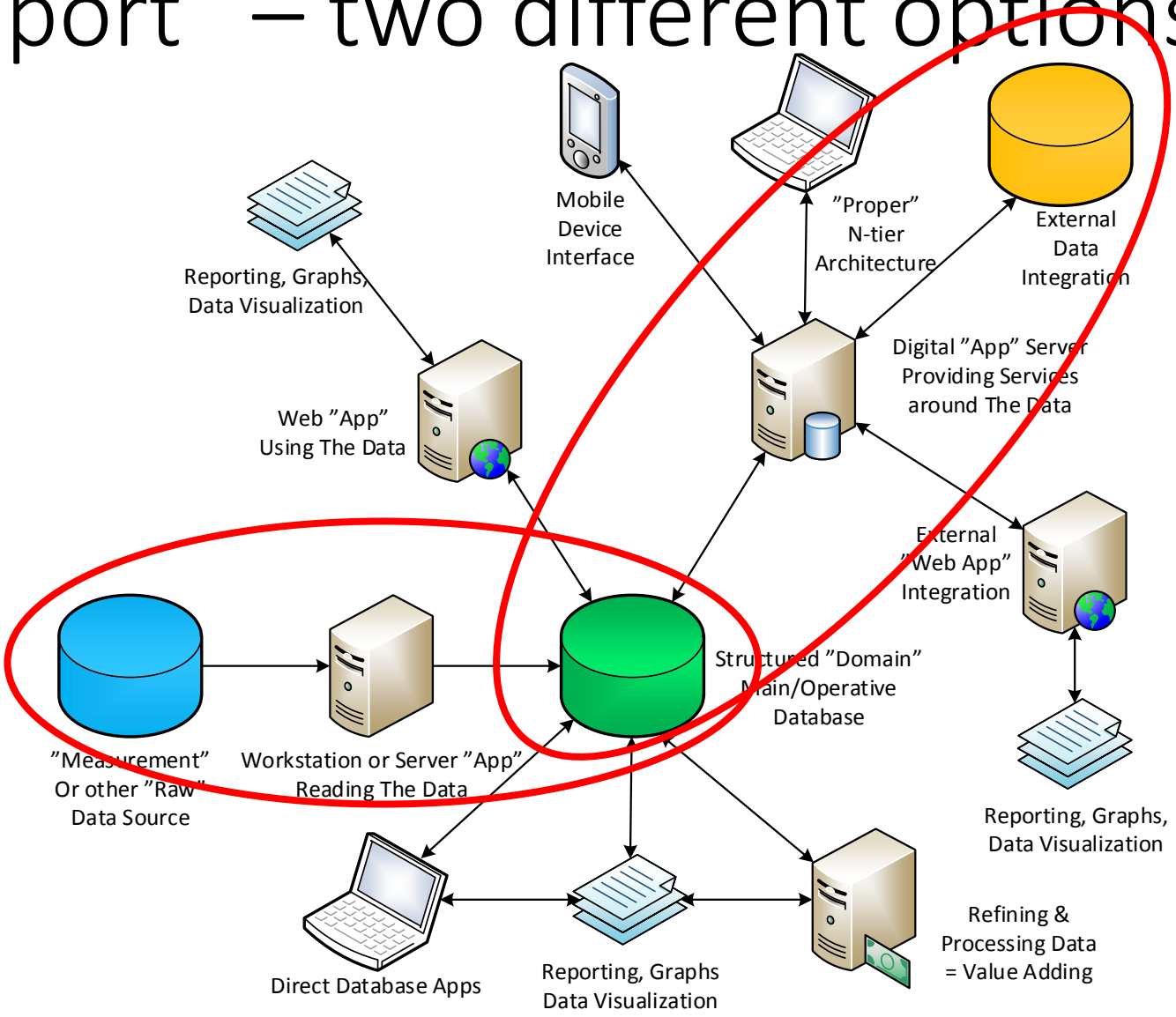
Technical vs. Logical Approach

- Technical approach cannot solve reusability in practice
  - Different solutions share technical constructs in a manner that they cannot be reused
- Import/Export, Database, User interface
  - Digital Service Interface = combination of above
- Logical approach makes clean distinction
  - UI is a digital interface with human usable design
  - Recognizing software development needs allows accelerating the development in a highly reusable manner

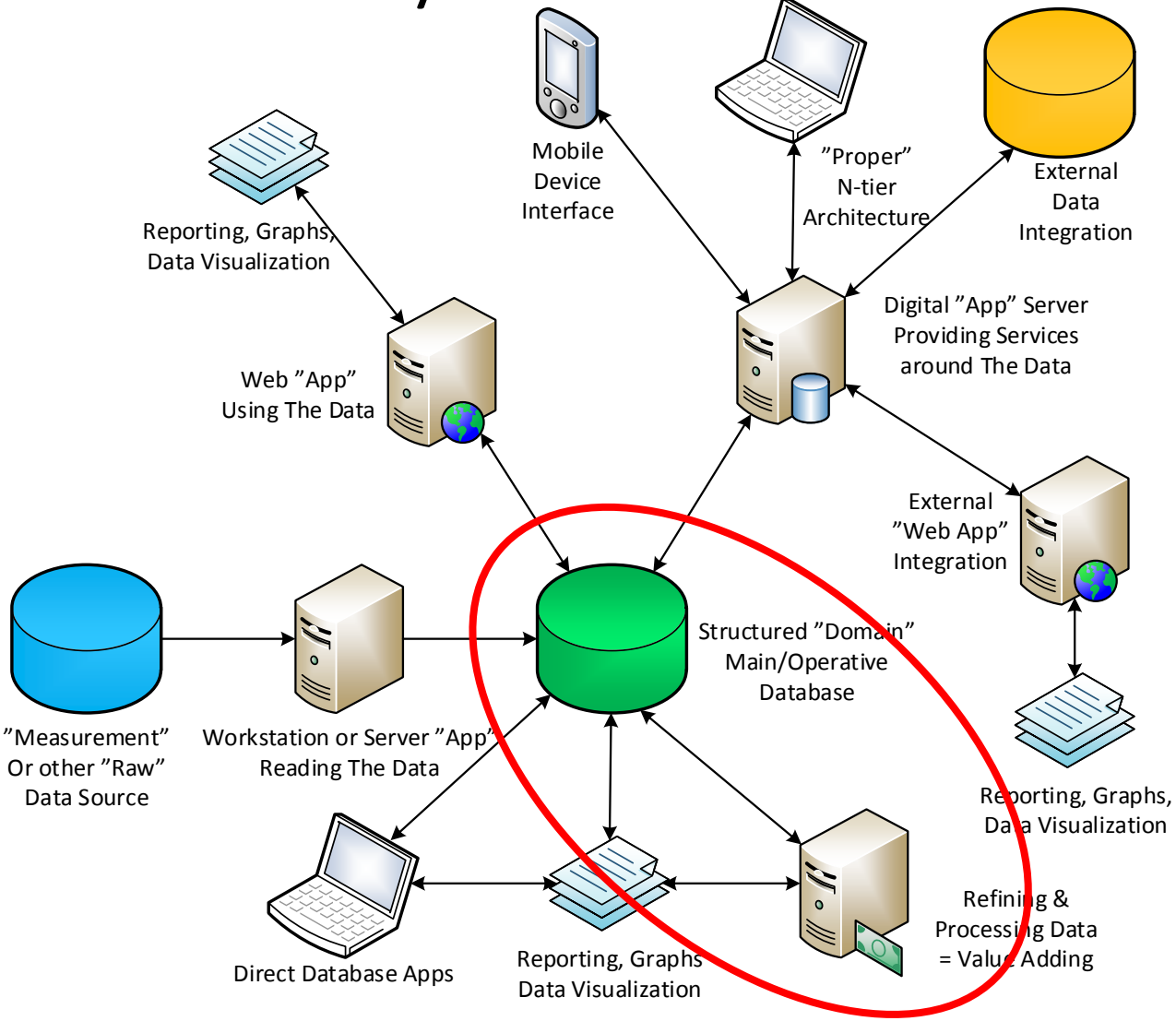
Multiplied Work = Multiplied Cost, including the whole lifecycle



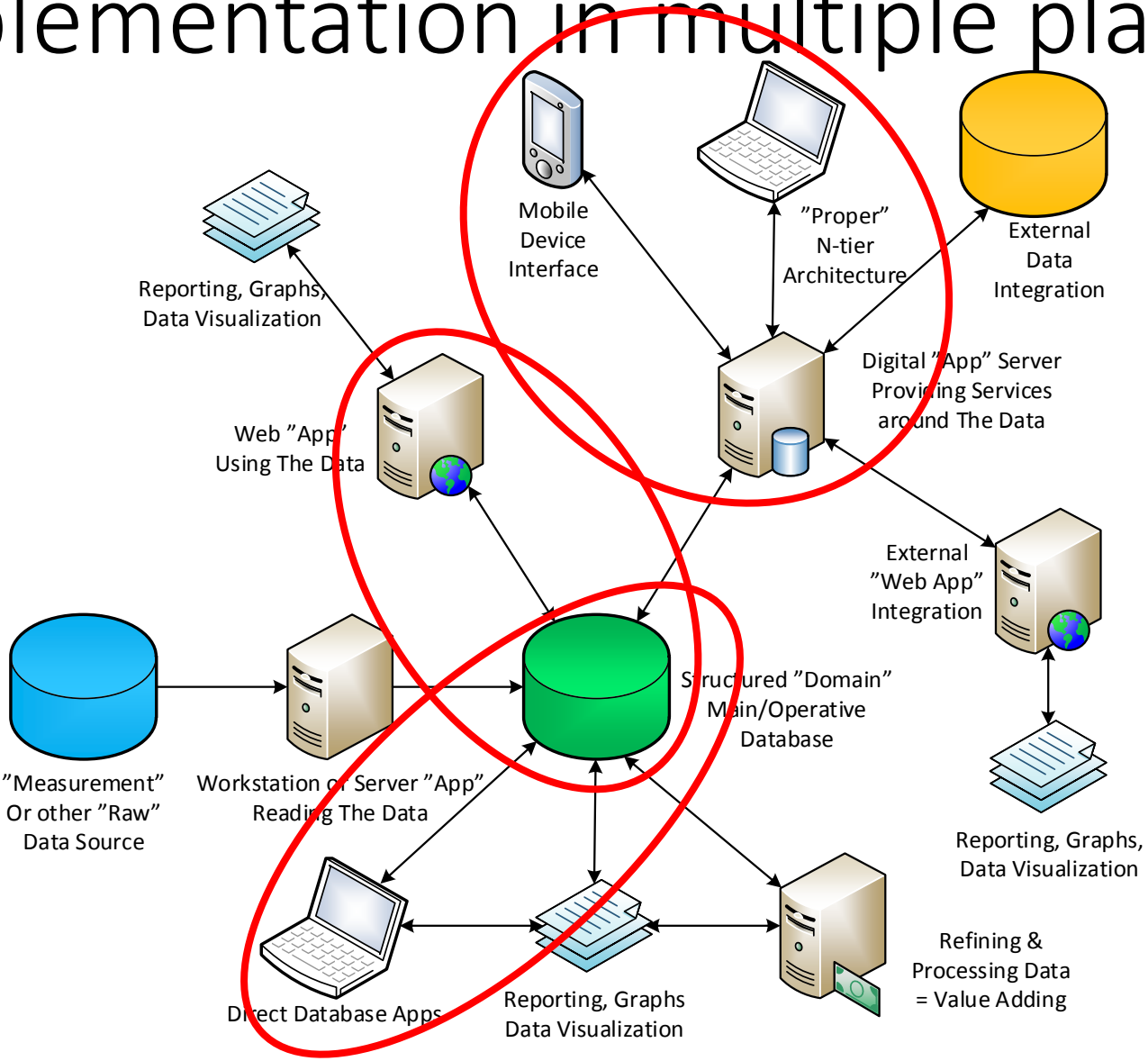
Example: ”Internet-of-Things data Import” – two different options



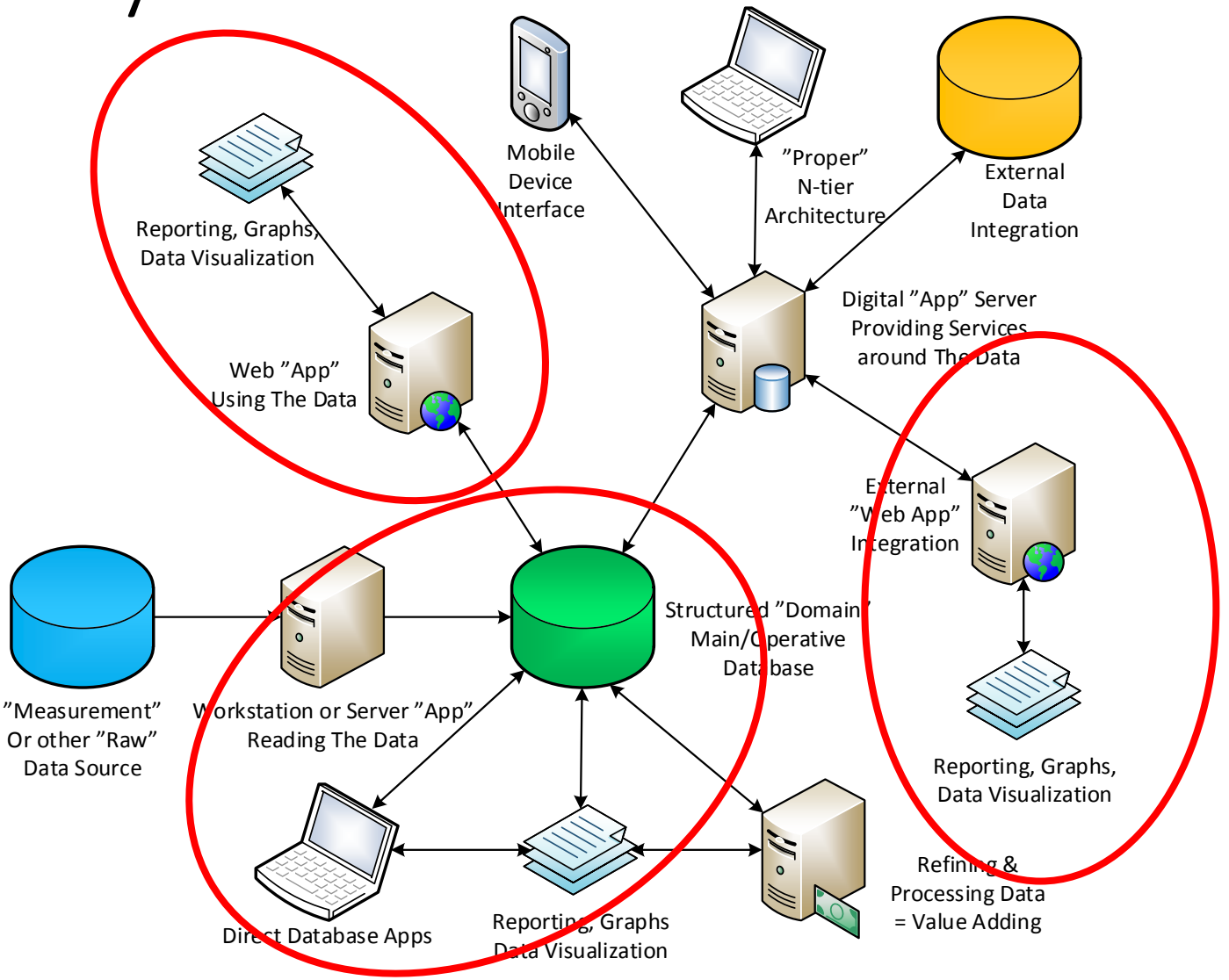
Data processing, restructuring = exists in every solution



User Interface technical implementation in multiple places



Visualization, analytics, reporting = always ”custom” to the solution



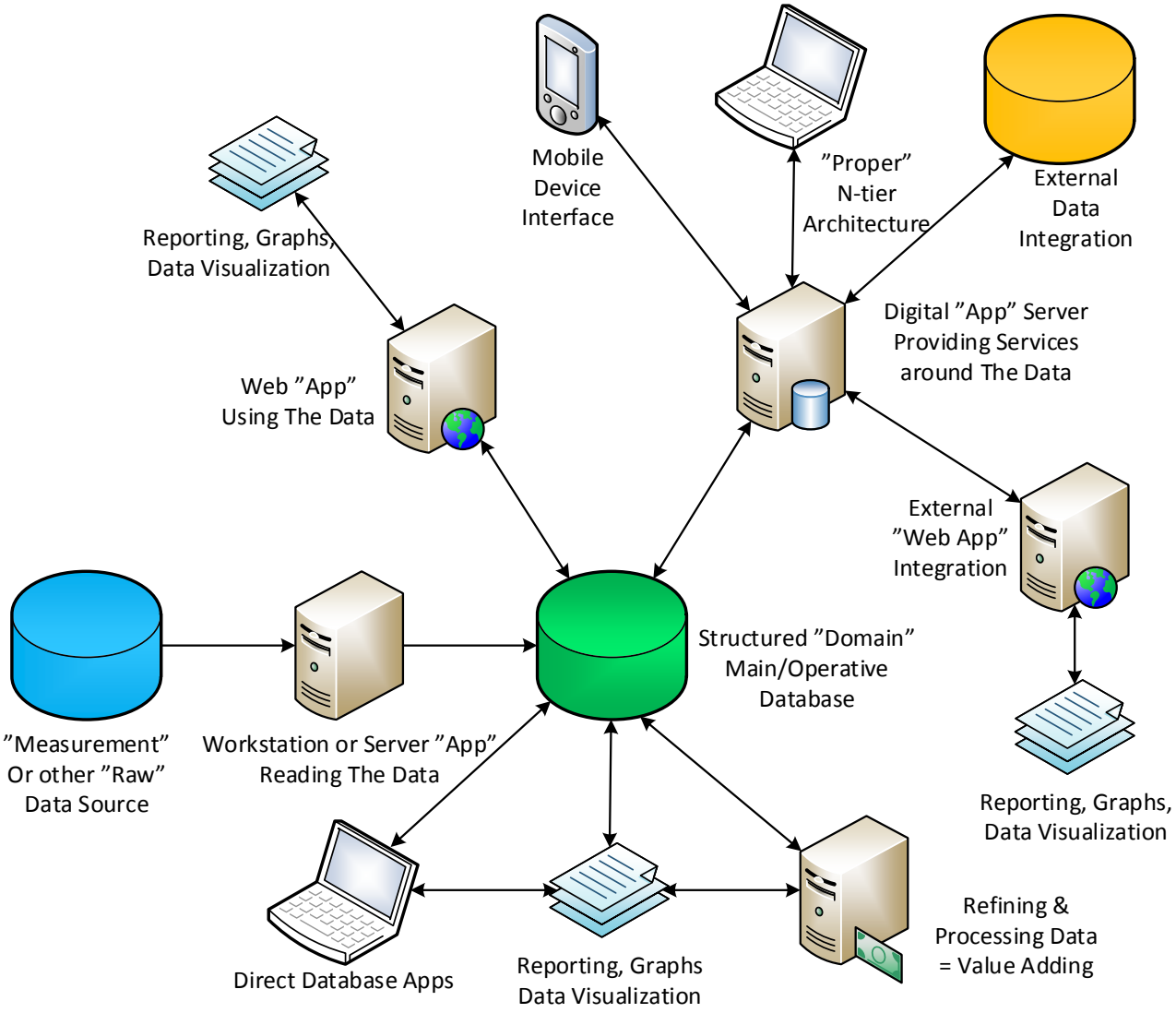
”Logical network”

Technical solution combined from logical elements

Solution is always a combination

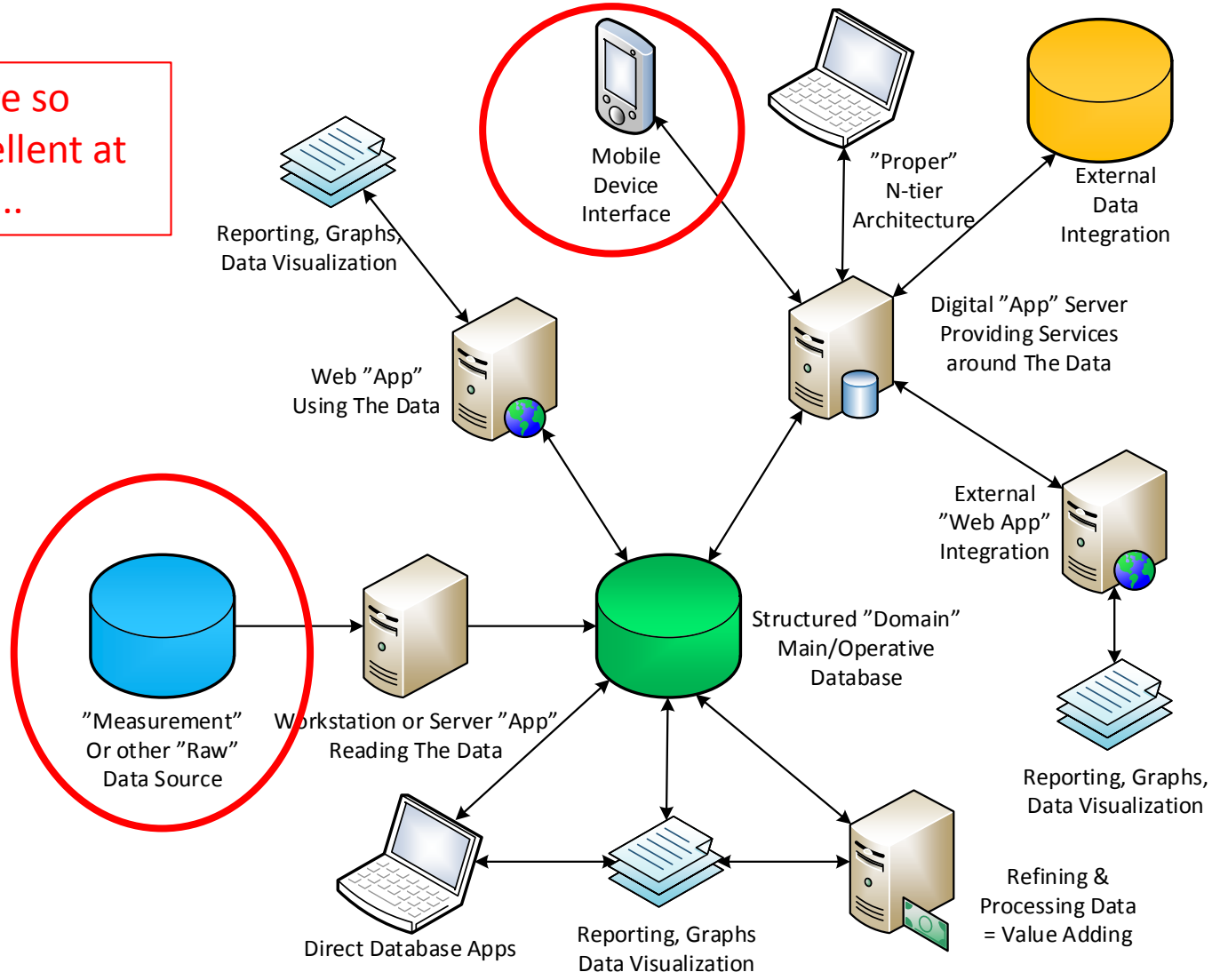
- Complete solution needs combination of elements
- Even single company/provider has logical overlapping
- Specialized focus of expertise is difficult to realize
- ”Were world experts on this area”
  - Implies (too) often the oversight, that you might be below average on the ”not-our-specialty” area

Example: Mobile-device visualization for measurement data

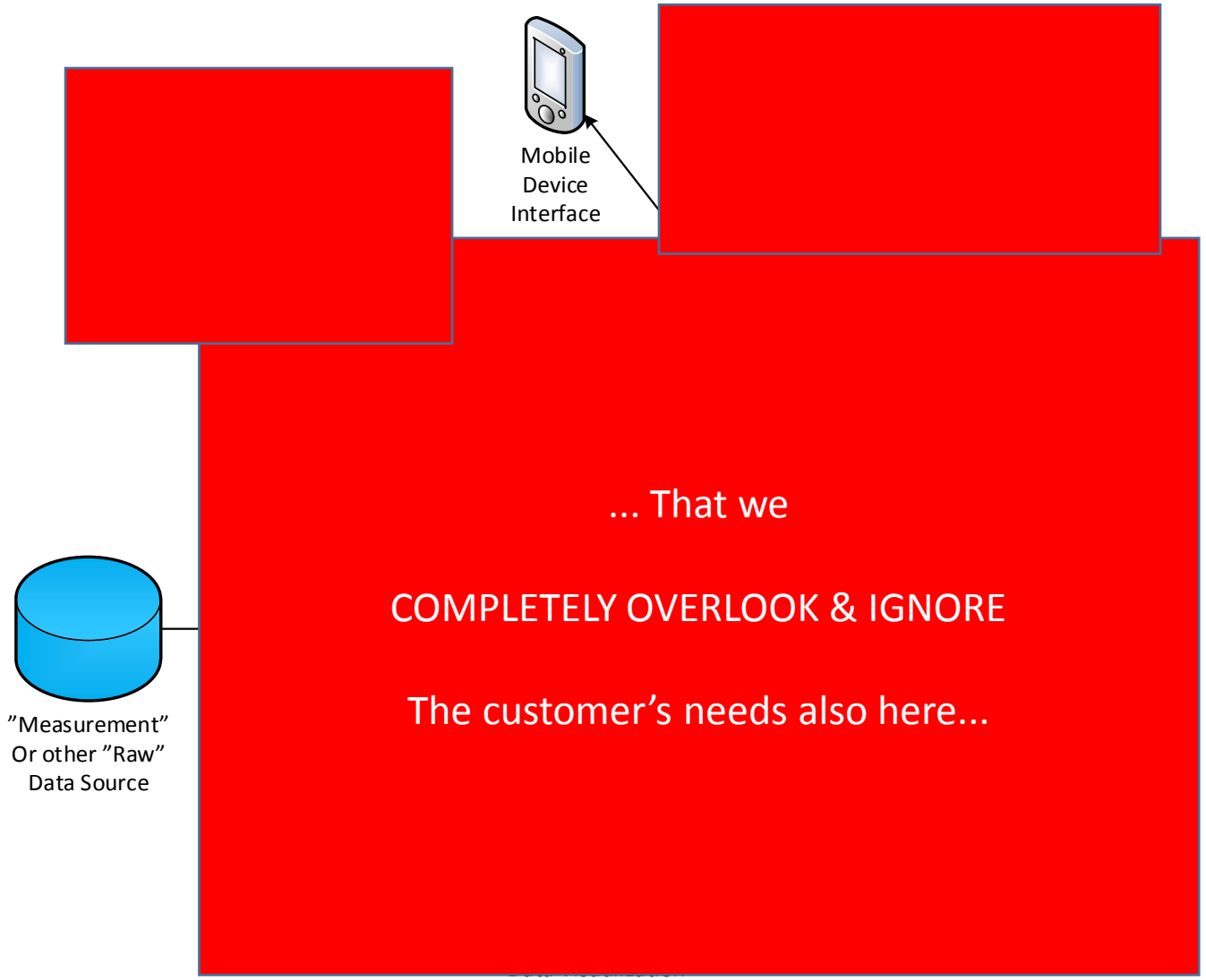


Example: Mobile-device visualization for measurement data

Were so excellent at this...

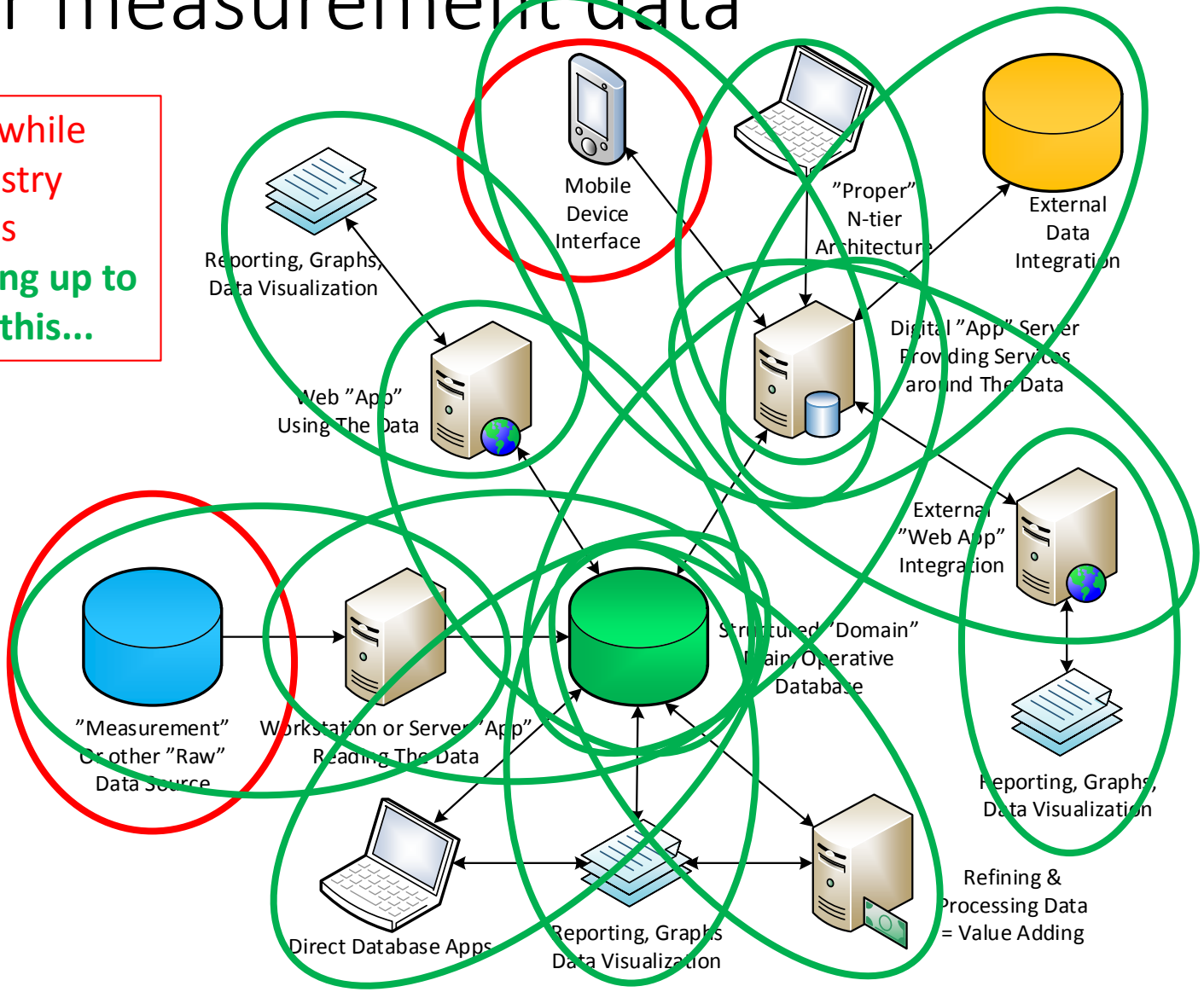


Example: Mobile-device visualization for measurement data

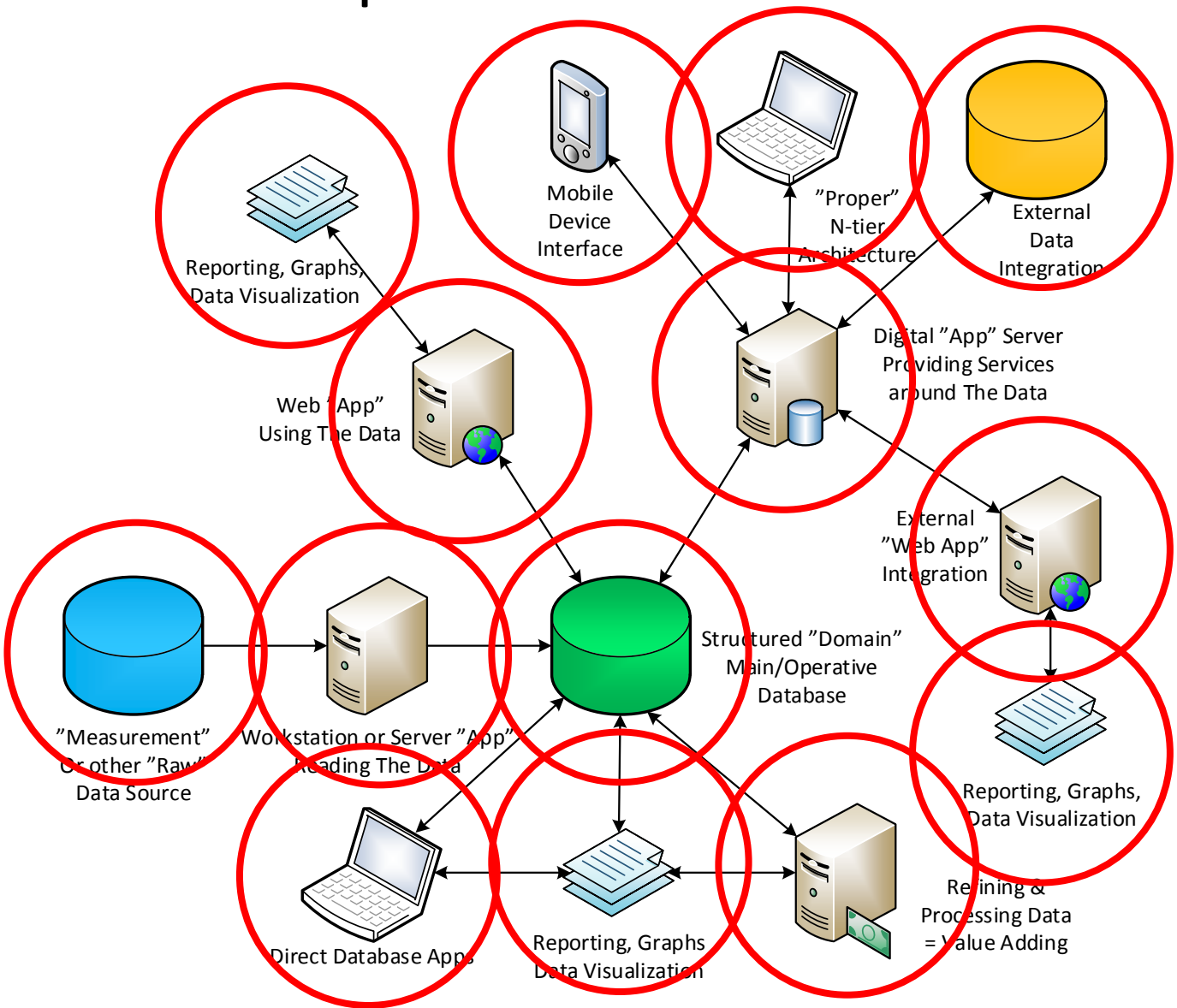


Example: Mobile-device visualization for measurement data

... Meanwhile our industry vertical is **partnering up to provide this...**



Focused expertise and leadership



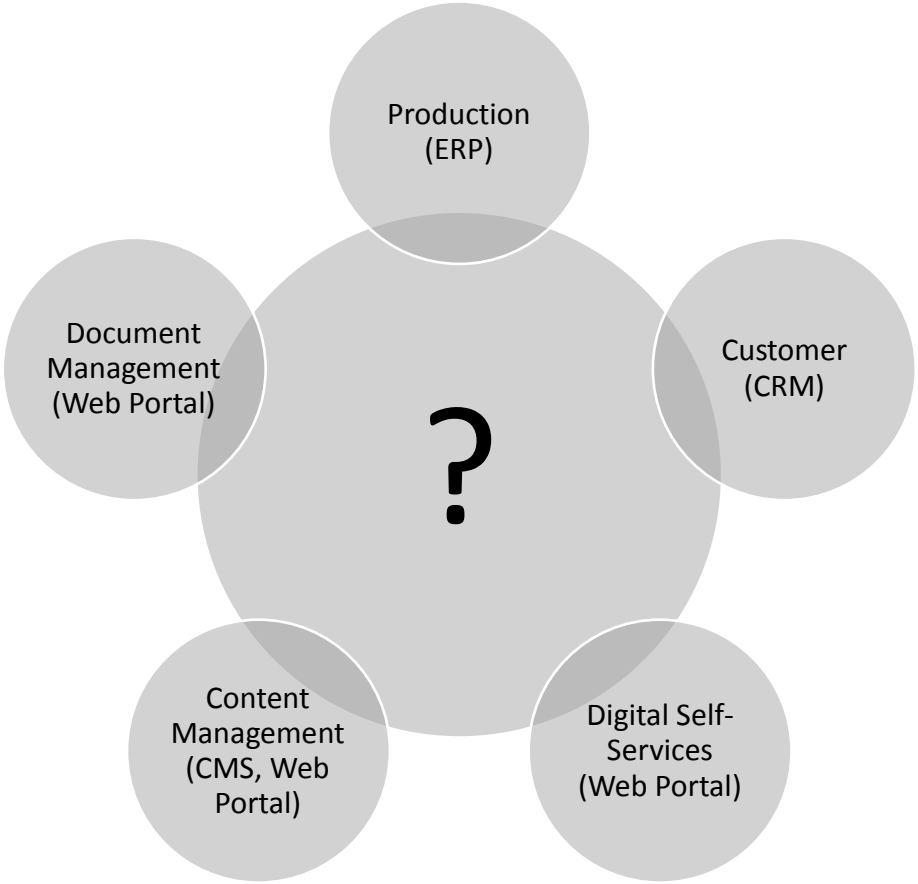
Tech leadership of ”the fittest”

- Cost structures favor flexibility
  - Unjustified work/cost is simply abandoned in the picture
- Transparency and efficiency of the network is the key
- First movers and early adopters paint the road
  - If you lack tech competency, partner up
  - If you feel like competing, what you’re competing against?
- Open platform, decentralized, available for everyone and for everything

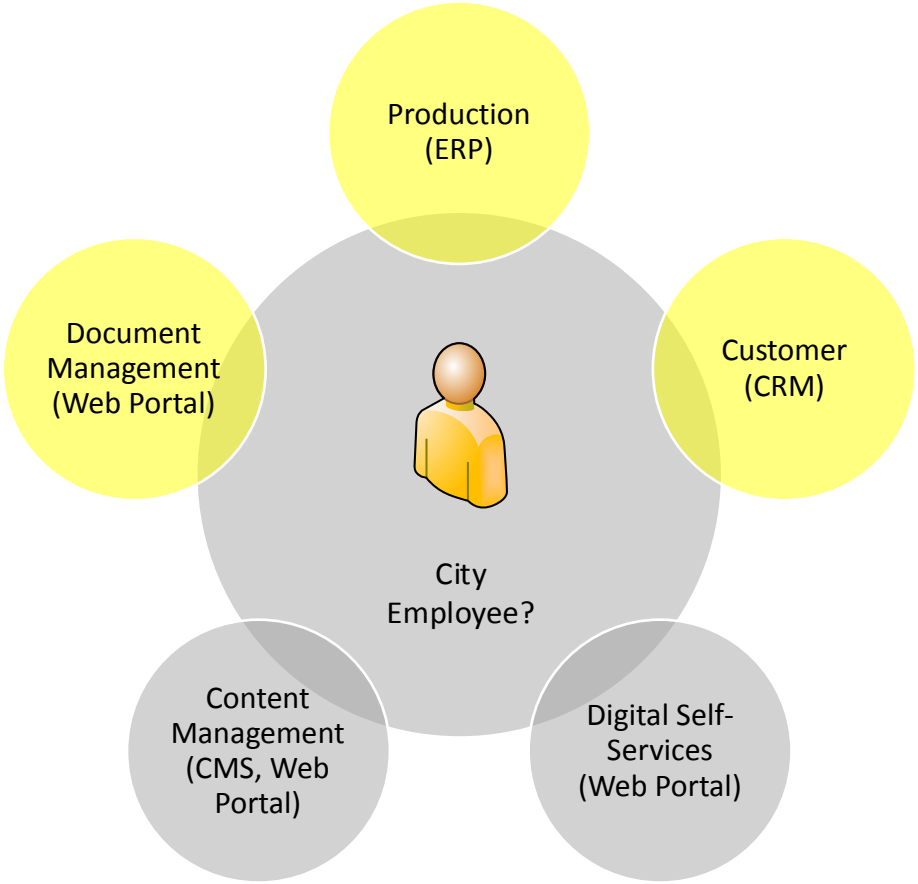
Scenario: City’s Services

Traditional ICT is ”siloed”

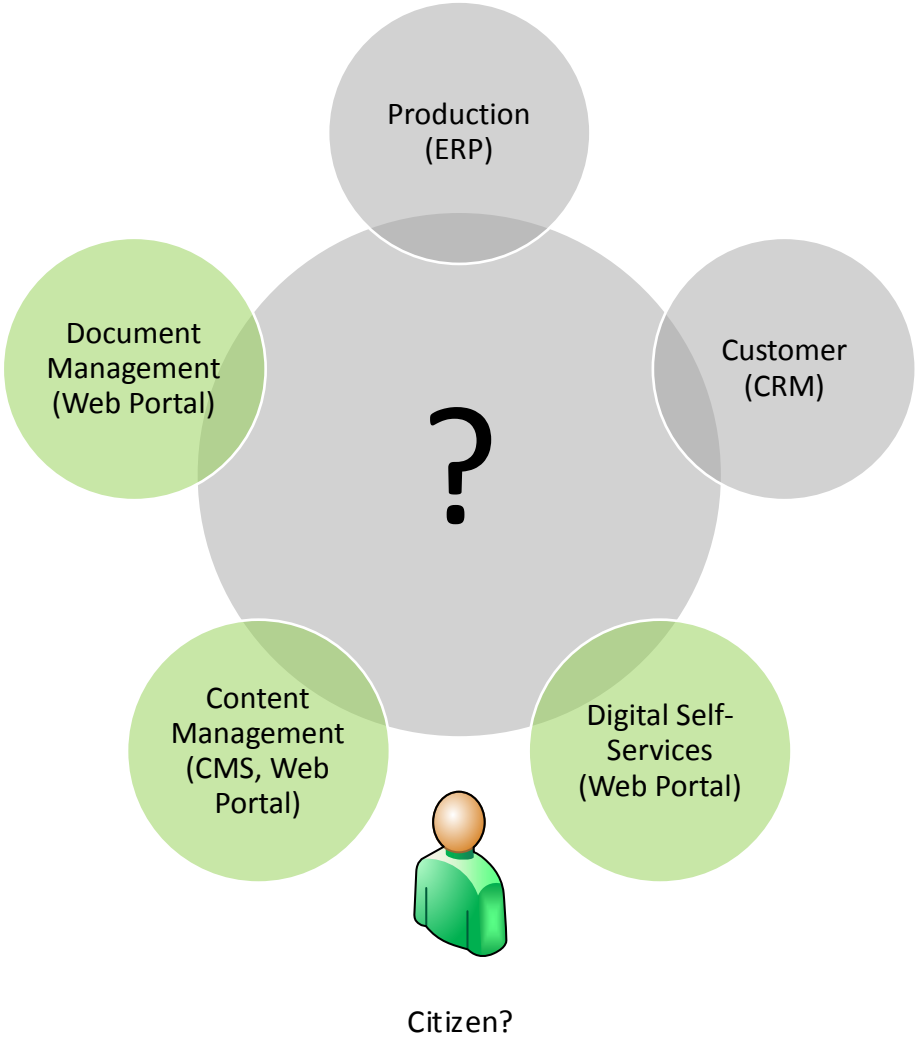
Problem: System centered perspective



Employee's perspective?



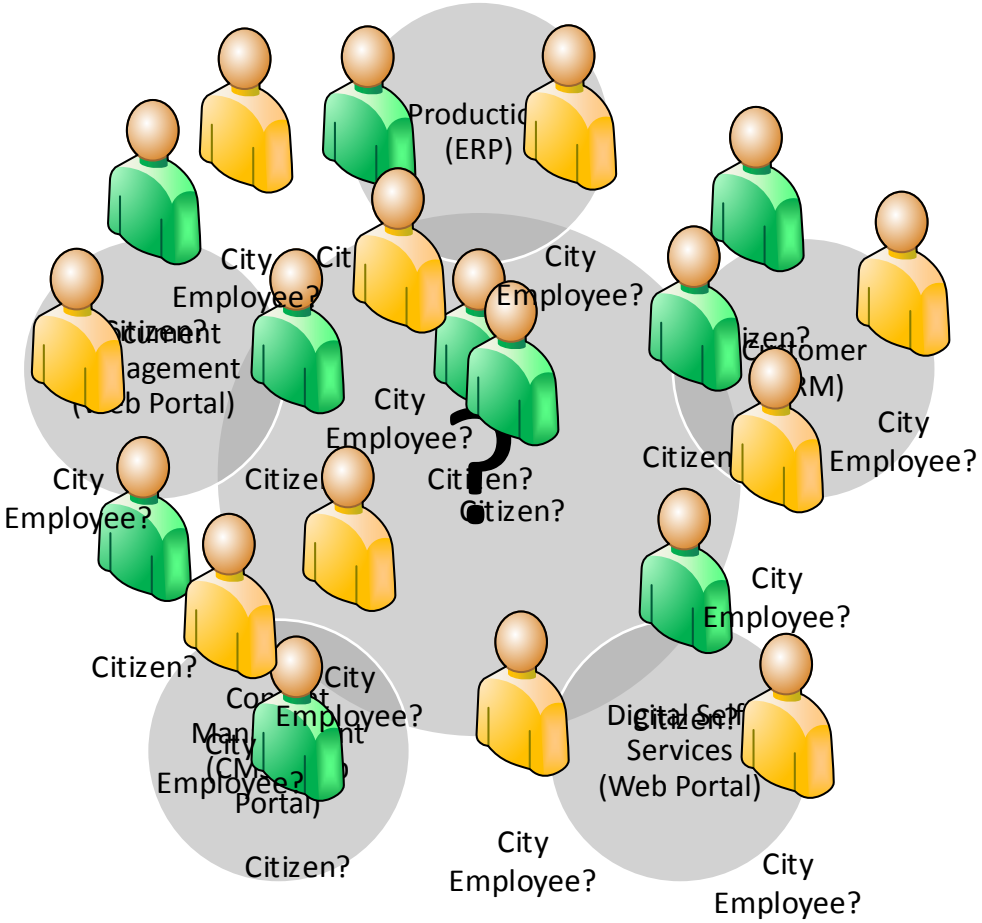
Citizen’s perspective?



City's Services are for Citizens

- City employee has multiple roles as providing city's services
- City employee is also a citizen of a city
- Citizen has multiple roles as requiring/requesting the services

Siloed systems don’t recognize roles, even less the combination of roles



Change

Perspective adjustment towards the true role of city

Perspective change: from ”user role” towards real world citizen

- Perspective needs to be individual citizen bound
- Citizen is using and providing services in multiple roles
- Citizen’s all personal roles have the impact of available services and the personal view of those services

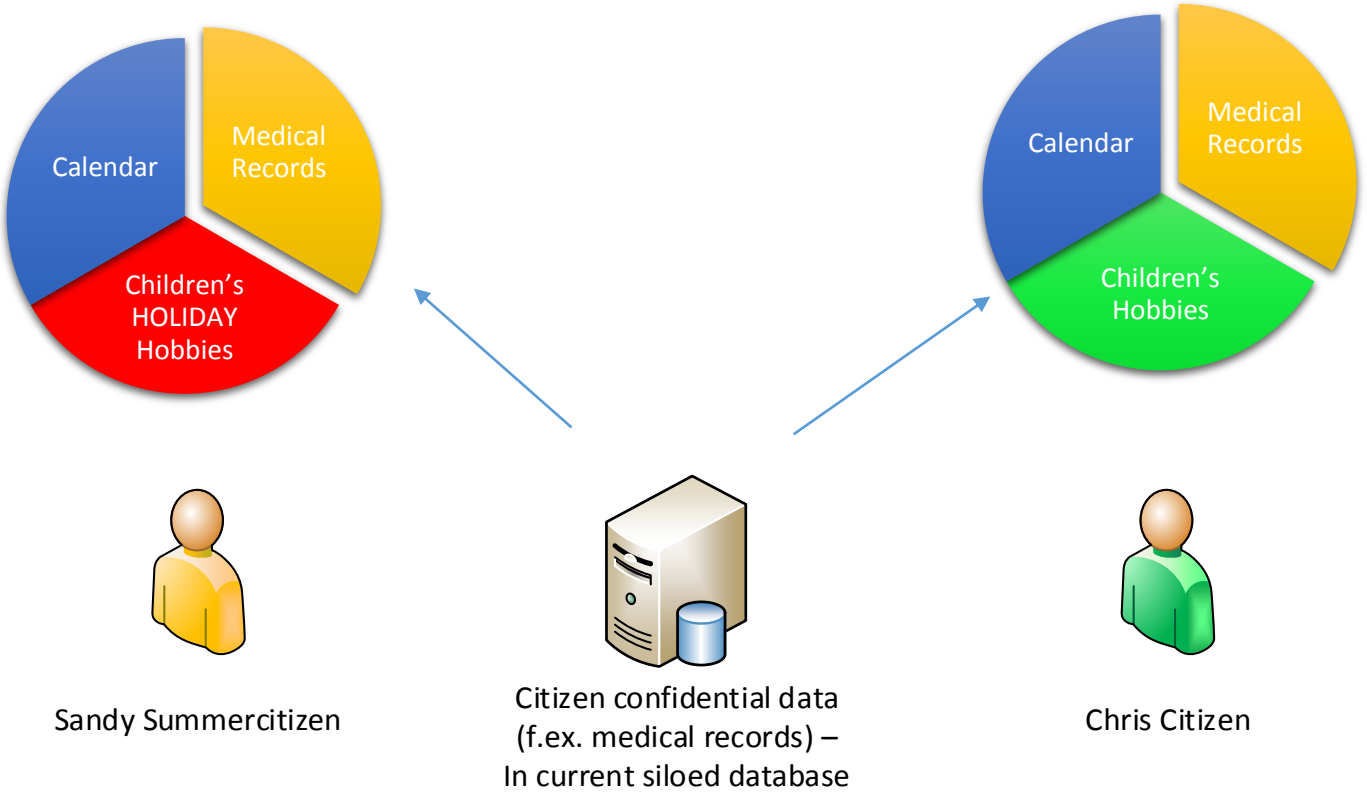
Nobody is just a citizen

- Private sector services MUST be in the ”view”
  - Not only the ”privatized” public services, but whole sector
- Non-profit, NGOs MUST be in the ”view”
- Everybody/everything MUST be allowed into the ”view”
  - Innovation is blocked if subjective selection is applied

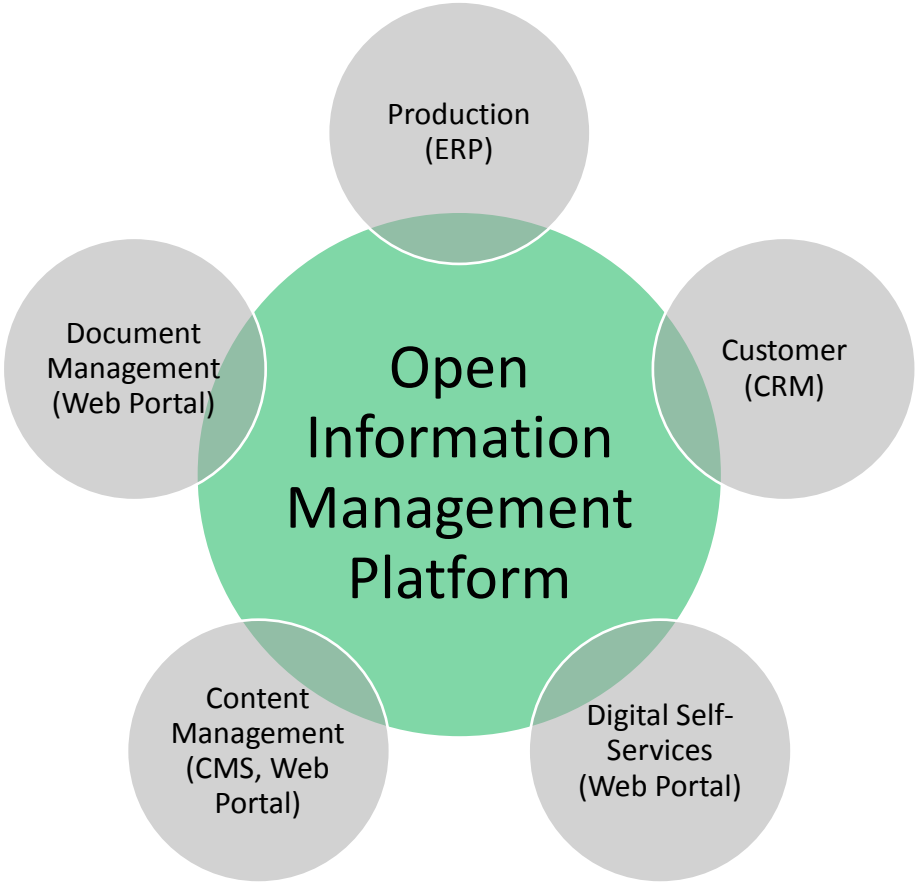
Means & **Benefits**

The city role for its citizens opens up and expands

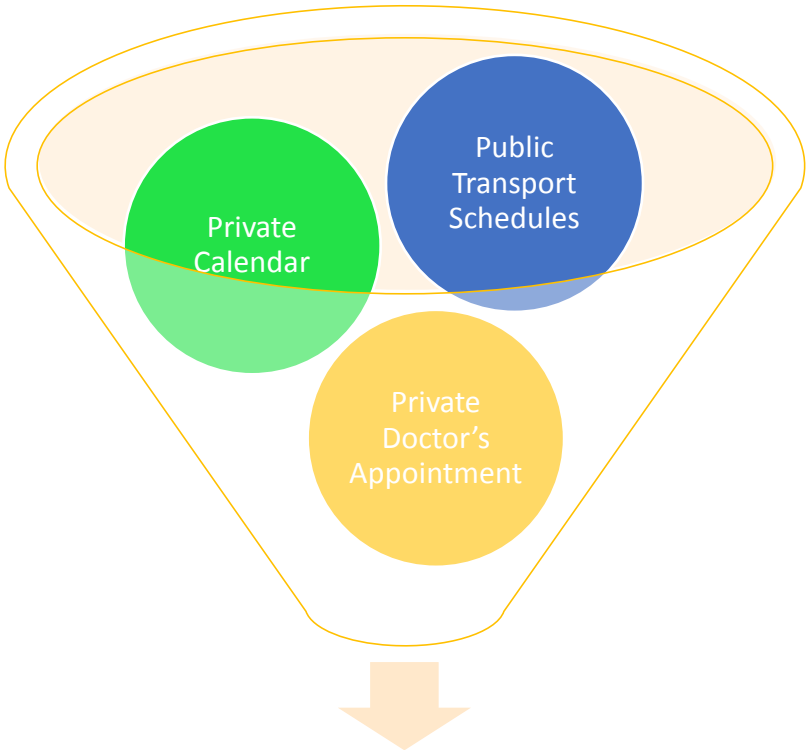
Citizen's data belongs to that citizen



Migration from current siloes



Individual citizen ”view”



Benefits of such an open information platform

- Differentiating factor: **Person owns the information** – he or she decides which ”service apps” he wants to run against that information

- Open, neutral party auditable platform covers the privacy and security
  - ”Apps” can be made by anyone – they’ll be ran in secured sandbox, so they don’t leak information or corrupt it
  - Think like open data interfaces: except that interface creation/modification is rapid – same as ”apps”

- City’s own need to produce/pay for ICT-solutions is dramatically reduced
  - **The cost drops to fraction**, when city services are served in parallel with the private sector offerings and private sector takes care of the customization of ”views”

- The technical framework of the platform is very light
  - The elements exist in current investments and projects – only the perspective change is missing
  - **The private sector creates the platform** by simply integrating their offerings to perspective model – the public sector just needs to mandate the possibility

Platform's model for "Apps" favors small (also local) ICT-companies

- Open development spreads – and develop also elsewhere to also return the benefits from elsewhere
- Early adopters and leading ICT-companies get to benefit from local city "familiar ground" of problems
- Outside bound advanced ICT-companies can also penetrate to the market and benefit the citizens
- Locals can become channel partners to close the gap

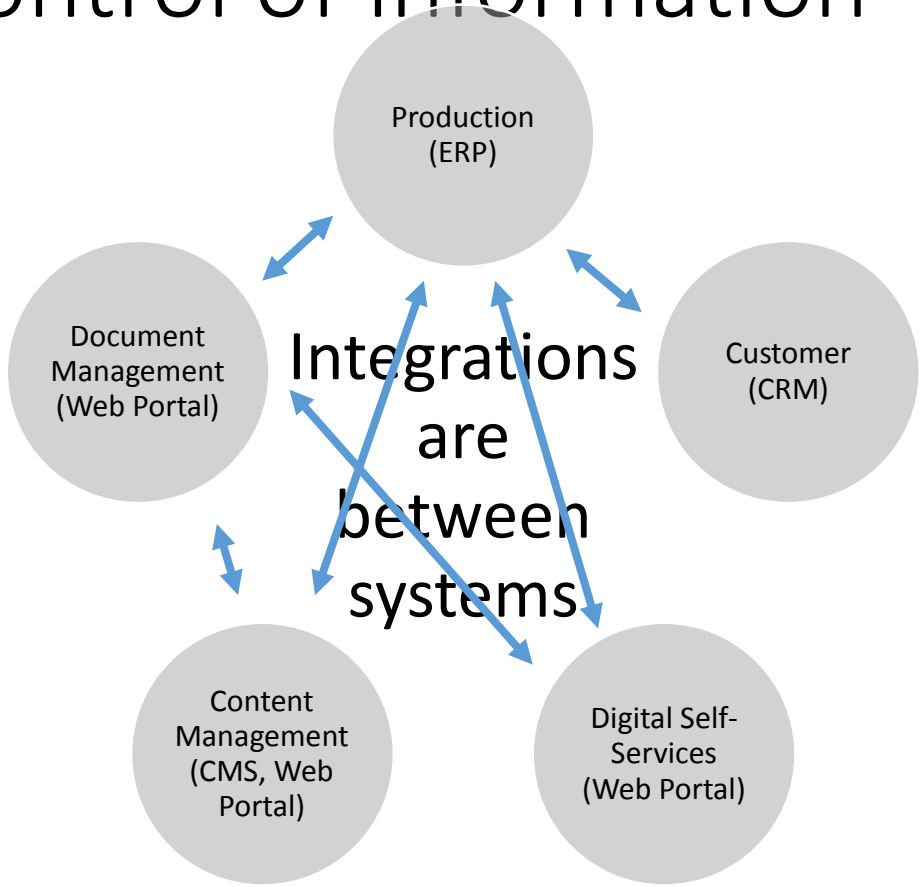
End of City scenario

Migration to ”The Ball”

From integration through service network to ”The Ball”

... Straight to ”The Ball”!

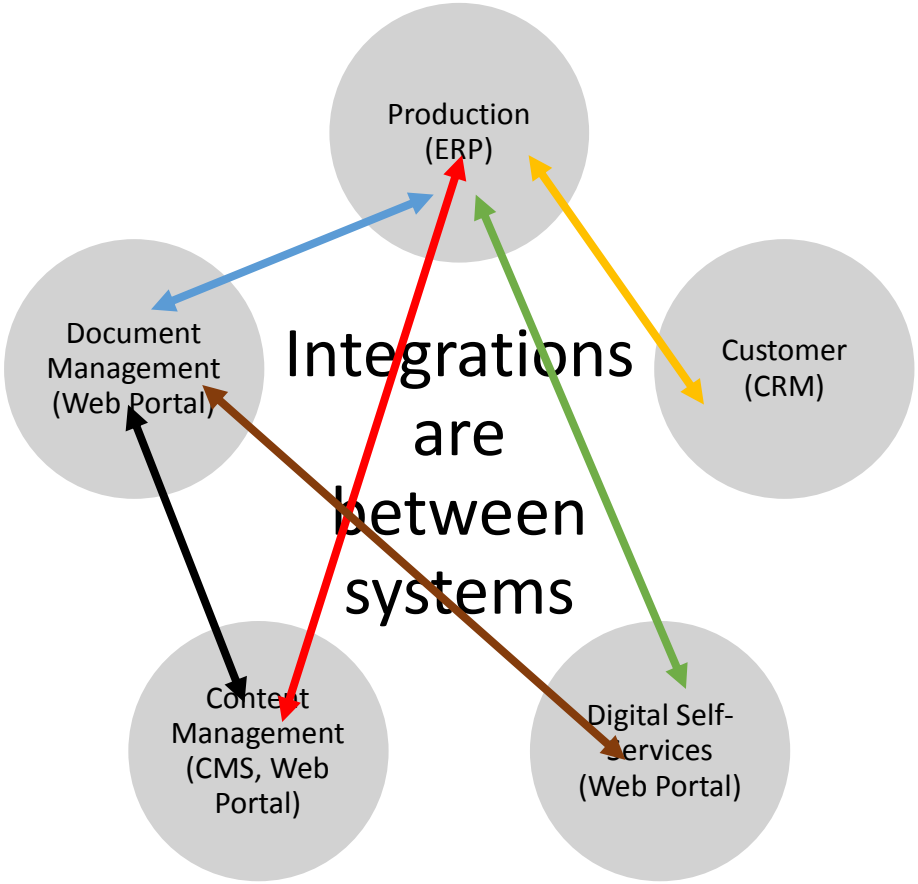
Traditional integration is technical; not in control of information



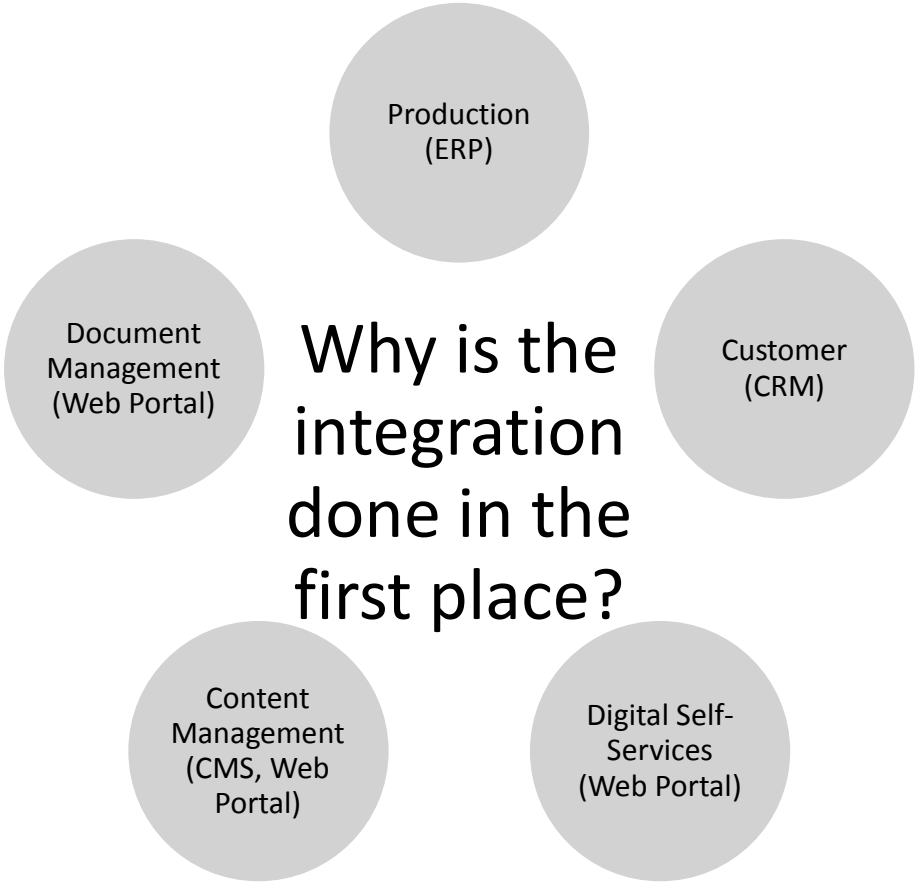
Traditional Integration Process

1. Design and decide which information needs to be integrated
    1. Document the decisions to specification
2. Design the logical and technical contracts
    1. Direction of calls, who calls, who serves - on which part of the interaction
    2. Specific names of services, service methods, parameters, return values
    3. Agree on common technical protocol(s) to use
    4. Document the decisions to specification
3. Implement according to specs (made above), either by ”coding” or ”configuring”
    1. Configuration is option on some sophisticated service stacks, works usually much better on paper than in practice
4. Maintain both ends simultaneously (changes applied in compatible manner)
    1. If the implementation is made in bundled manner (applies often for configuring approach), requires larger stack to be maintained at single bundle = much heavier coordination and syncing
5. Implement custom aspects system-specific (= often manual coding) when facing differences between systems – about authentication, users, information ownership or any other difference
    1. No common maintainability perspective, always case specific, often also original provider/producer-specific
    2. Specialized systems are ALWAYS case-specific, underlining that the difference is always there – that’s why the specific systems are brought up in the first place (ERP for ERP, CRM for CRM, CMS for CMS...)
    3. Integration applies always down to database storage and domain logic rules - risks for compromises under pressure
    4. (Document the implementation by updating the specification accordingly – not reality in all the cases)

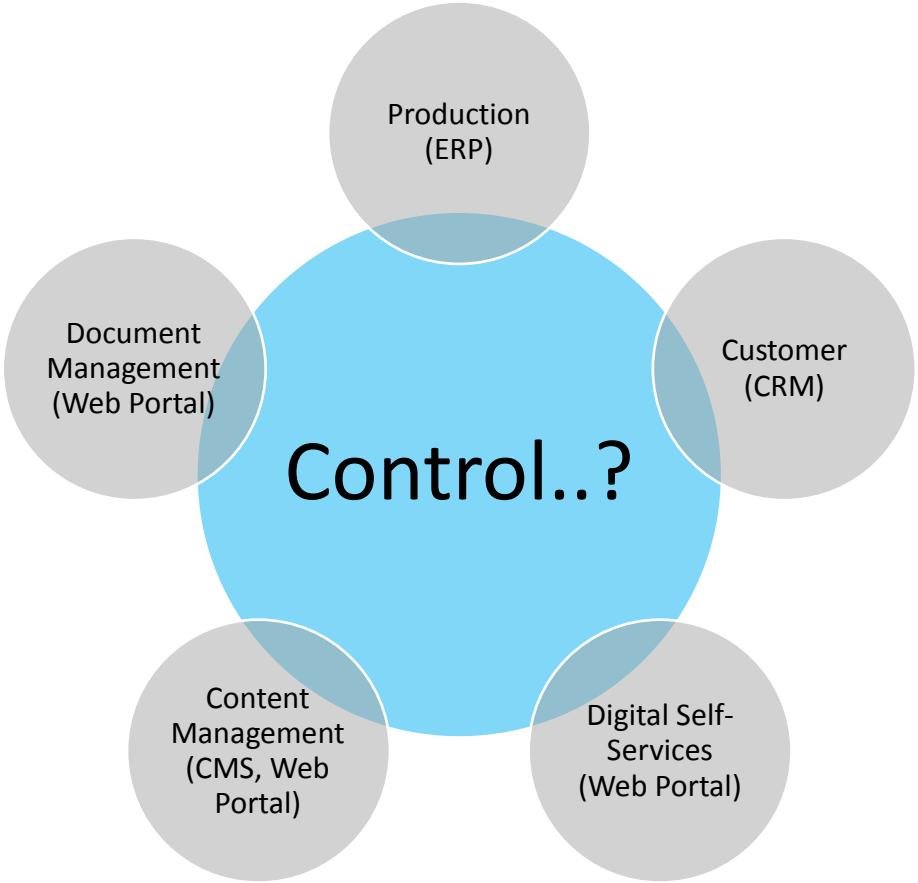
Integrations are often unique, to the data and business domain rules



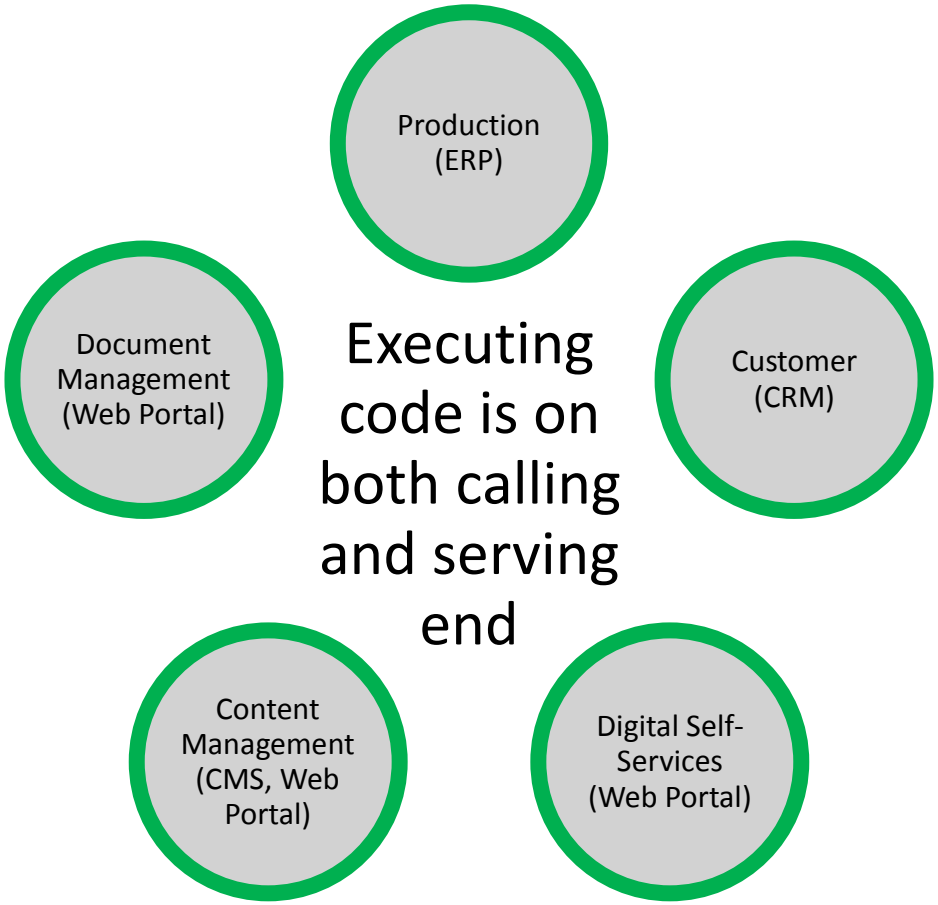
STOP!


Production (ERP)
Document Management (Web Portal)
Customer (CRM)
Why is the integration done in the first place?
Content Management (CMS, Web Portal)
Digital Self-Services (Web Portal)

To automate larger joint system, to control information in big picture..?



The actual integration is implemented always in interfaces



ADM-Powered Integration Process

1. Design and decide which information needs to be integrated
    1. ~~Document the decisions to specification~~ Define on logical level => generates also specification documentation
2. Design the logical ~~and technical~~ contracts
    1. Direction of calls, who calls, who serves - on which part of the interaction
    2. Specific names of services, service methods, parameters, return values
    3. ~~Agree on common technical protocol(s) to use~~
    4. ~~Document the decisions to specification~~ Define on logical level => generates also specification documentation
3. ~~Implement according to specs (made above), either by "coding" or "configuring"~~
    1. ~~Configuration is option on some sophisticated service stacks, works usually much better on paper than in practice~~
4. ~~Maintain both ends simultaneously (changes applied in compatible manner)~~
    1. ~~If the implementation is made in bundled manner (applies often for configuring approach), requires larger stack to be maintained at single bundle = much heavier coordination and syncing~~
5. ~~Implement custom aspects system-specific (= often manual coding) when facing differences between systems – about authentication, users, information ownership or any other difference~~
    1. ~~No common maintainability perspective, always case specific, often also original provider/producer-specific~~
    2. ~~Specialized systems are ALWAYS case-specific, underlining that the difference is always there – that's why the specific systems are brought up in the first place (ERP for ERP, CRM for CRM, CMS for CMS…)~~
    3. ~~Integration applies always down to database storage and domain logic rules - risks for compromises under pressure~~
    4. ~~(Document the implementation by updating the specification accordingly – not reality in all the cases)~~
6. Implement code within strictly controlled ADM-automated reference architecture blocks
    1. Unified maintainability, unified over whole network, not personalized/provider locked
    2. Identify automation candidates of repetitive patterns even further when developing down the road

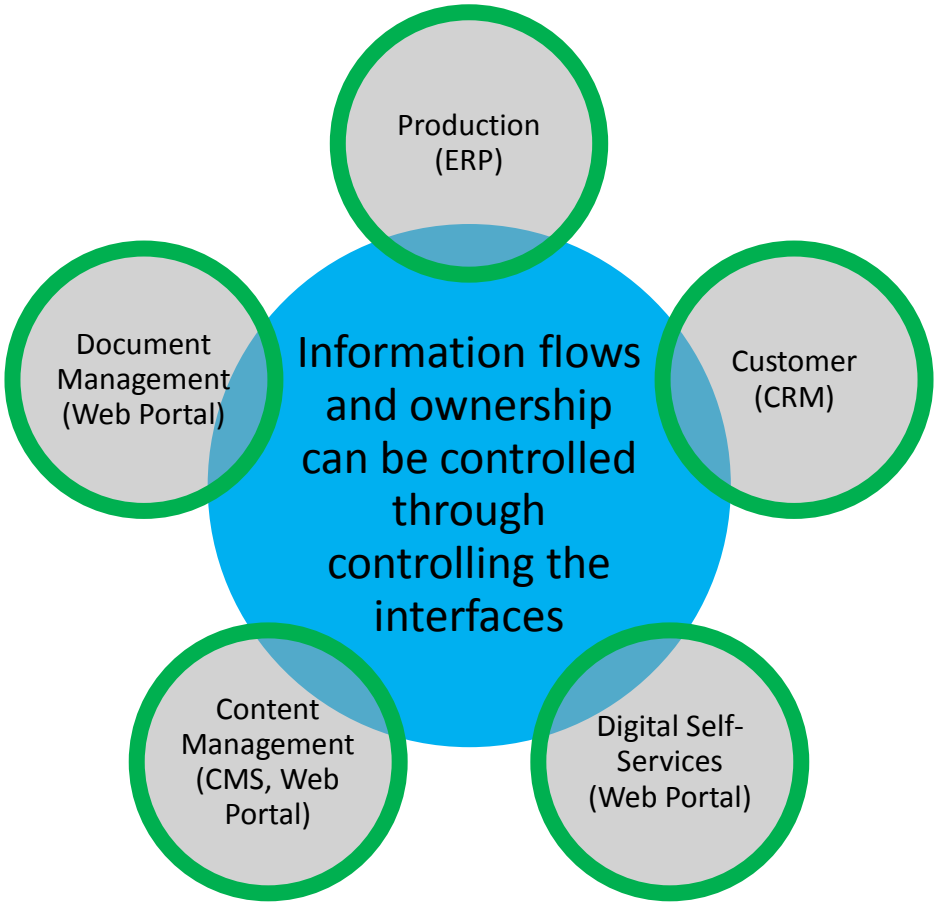
No need to implement anymore?!

- Implementation is needed ”first time”
  - In reusable, reference architecture best practice manner
- Technology-platform specific needed once per platform
  - Replaces around 80-90% of the manually maintained code
- Requires mainteinance and updates as manual coding...
  - ... The changes/updates target the **WHOLE NETWORK** at once

ADM-Powered Integration Process

1. Design and decide which information needs to be integrated
    1. Define on logical level => generates also specification documentation
2. Design the logical contracts
    1. Direction of calls, who calls, who serves - on which part of the interaction
    2. Specific names of services, service methods, parameters, return values
    3. Define on logical level => generates also specification documentation
3. **If the required technology-platform is missing the implementation**
    1. Implement reference best practice in traditional and manual manner
    2. Convert the reference into ADM-automation => code and artifact generators
    3. Publish the addition/change/update to network source code repository
4. Implement code within strictly controlled ADM-automated reference architecture blocks
    1. Unified maintainability, unified over whole network, not personalized/provider locked
    2. Identify automation candidates of repetitive patterns even further when developing down the road

Logical level information management is enabled…



Service Libraries

Realizing distributed service chains or so called ”service bus”

From technical minimalism towards semantic minimalism

- Techical minimum for implementing digital service
  - The name of the service
  - Technical parameters of the service
  - Technical return value of the service
- Semantical minimum for specifying a logical service
  - The name of the service (the domain will appear as unique namespace)
    - Semantically MyApp.GetPerson vs. PopulationRegistrationOffice.GetPerson
  - Semantic definitions for parameters and for return value
    - MyApp.SocialSecurityNumber vs. PopuplationRegistrationOffice.SocialSecurityNumber
- Technical & semantical combined = Service ”thumbprint” or ”signature”
  - Service: PopulationRegistrationOffice.GetPerson
  - Parameter: PopulationRegistrationOffice.SocialSecurityNumber
  - Return value: PopulationRegistrationOffice.PersonInfo

ADM-Powered ”The Ball” Services

- Semantical specification and thumbprint are generated from ADM-automated logical level service definition
  - Unified model in defining services is natural path to catalogue/library => towards dynamic model
  - Registering deployed services to the catalogue/library is automated as well
- Migration path from current status of already running services
  - One big motivation is to make service library (= ”service bus”) usable
  - Logical definition is created from current technical level fact, filled up with specification details
  - Unifying the technical service protocol layer by applying ADM integration process (as described earlier)
- Creating new service from scratch
  - Logical level definitions mean less work compared to separate specs and implementation
  - End result is library / ”service bus” compatible and publishable service, that can still act as independent internal service as well

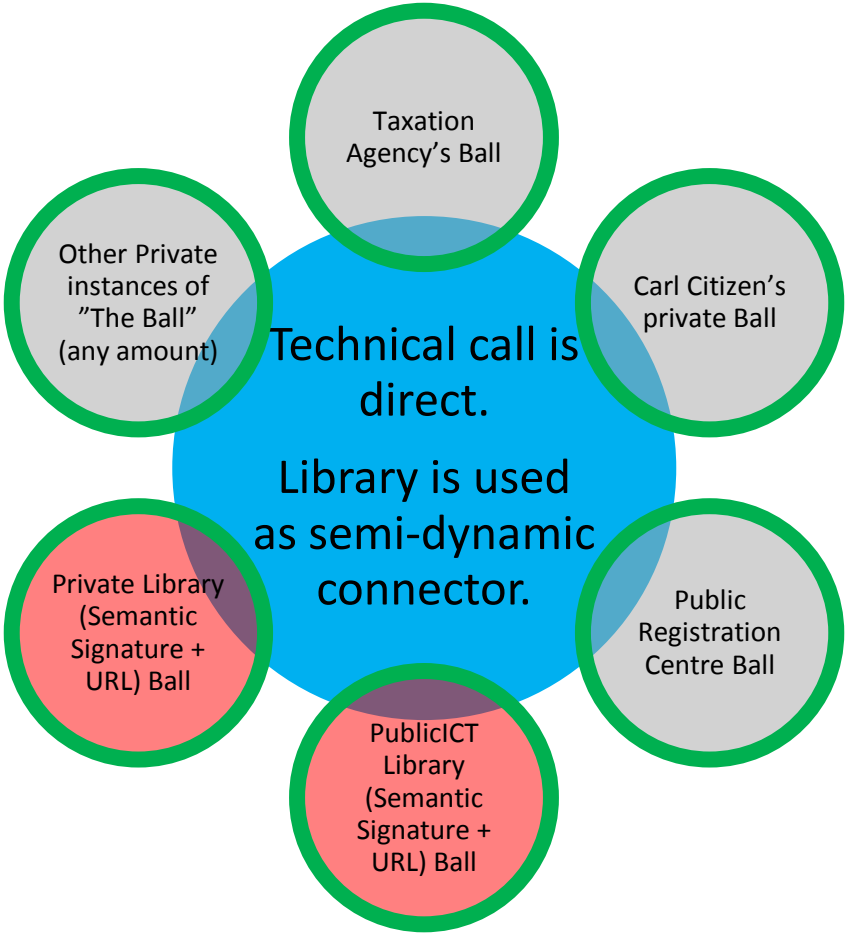
Library = Services among other services

- Catalogue is as distributed service as any other
  - Anyone can publish a library or catalogue service
  - Not every library has to be trusted – automation sure won't blindly trust everybody
  - The key is to be able to choose who to trust and who to not trust

- Searching the catalogue can be based on vertical-specific metadata as required
  - The compatibility of found services is the "semantical thumbprint/signature"
  - Private sector verticals define their own relevant metadata
  - Real-time embedded systems for M2M or IoT as one example of very technical critical requirements

- Minimum viable result from library search is complete semantic signature with URL address, where the actual service can be found/called from
  - Testing + demonstarting is straightforward to do with parallel URLs, service stubs, demo data, etc...
  - Concrete infrastructure can be built to support/accelerate the library use, but it's not necessary requirement – just required when expanding to non-technical information architects and normal people
  - The unnecessary heavy demo/testing is not required if the perspective is still technically skilled software developers and software architects

Technical actual call: semantic signature + URL where to call from



Establishing chains of services through library/catalogue

- MyApp internal service can take advantage of PublicRegistrationCentre (PRC) models and functional service blocks
  - Information models are used as PRC.Person – domain namespace models
  - Functionality runs as-is against the it's own domain models
  - The actual running instance of "The Ball" running by PRC is needed to secure the actual PRC controlled content – even if its still separated to citizen-specific areas
  - Functionality against citizen-specific data can also be ran by other instances, that the citizen authorizes to run – and where the citizen approves exporting parts of his or her data

- "Service App" is running other "service apps" against its own information/data storage
  - Semantic model is not bound to actual content from PRC. Semantic model just specified the data semantic structure.
  - Thus PRC:s data exported through secure channel onwards to Taxation Agency's systems are then refined further to TAX domain and TAX functionality

- **New Differentiating Feature:** Private sector services that comply with security and privacy
  - Even individual developer can build functional software that uses PRC.Person and TAX.TaxationInfo2011 models => "Service App" that can be ran from the library
  - Functionality of "Service App" is executed against citizen's private personal information within security bound, audited and proven private context
  - "Service App" with interface level control does not have to be a server in network – it can be block of code executing in same security-bound process. URL = Universal Resource Locator ☺

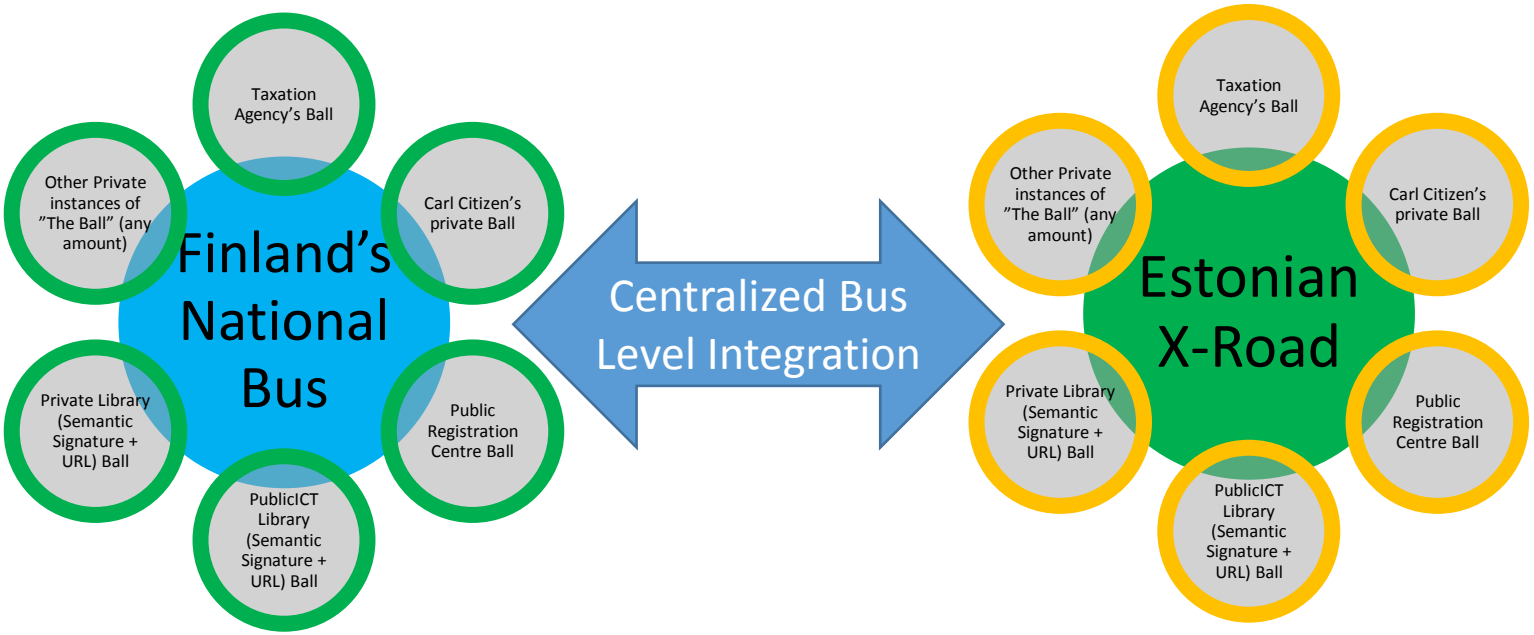
Integration of larger combination of systems

National Service Bus: Between different countries

Applies as a model within one country: Between different cities or government offices

... Universally between any combination of any sizes

Integration can be seen as integration of well known modeled systems…



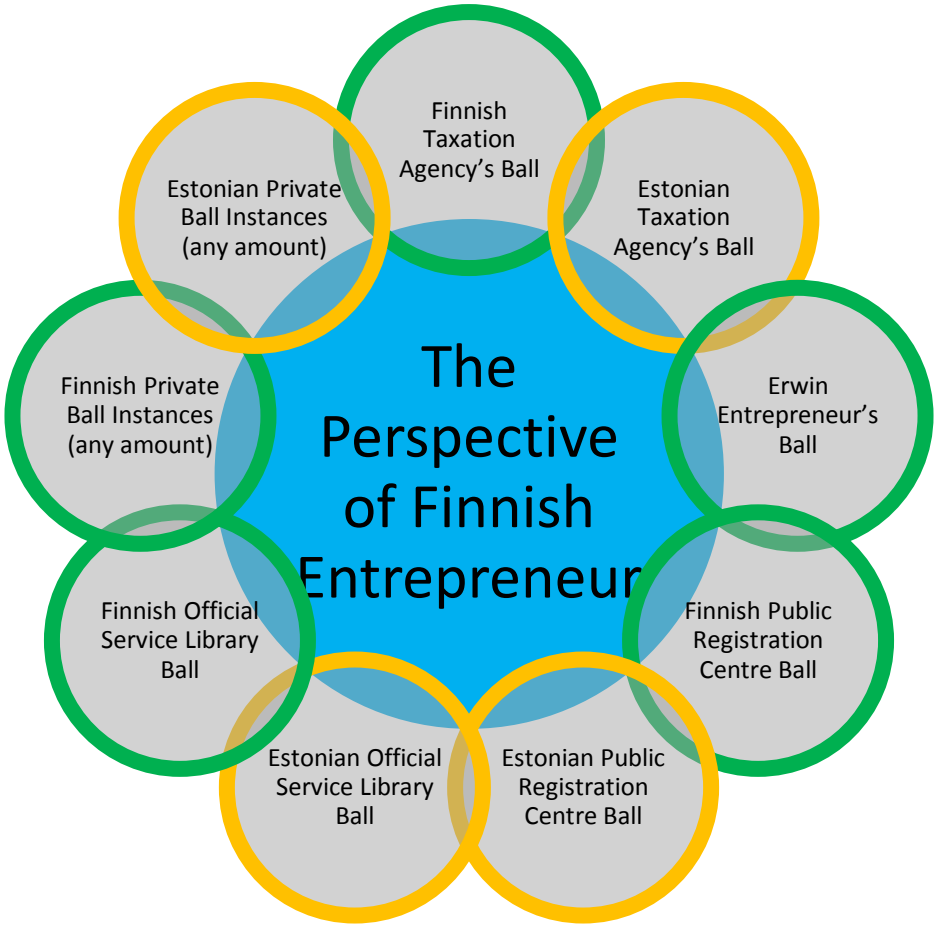
Distributed Model Removes Centralized ”Blocking of Progress”

- No need for centralized top-to-bottom dictate of matters
  - ”But Estonian doesn’t have The Ball based solution!”
  - Everything can be migrated to ”The Ball” – simply controlling the service layer as explained before
  - Everybody can make the migration simply wrapping the service layer – who’se to be trusted is completely different manner
  - = Finnish Ministry of Finance can wrap Estonian X-Road to comply with ”The Ball” architecture
  - = Finnish Ministry of Finance can publish trustworthy wrappings in their signed catalogue/library
  - Private and public sector actors can also choose to trust ”itself” and make all the above on its own – including the transparent auditing the trustworthyness of ”wrappings”

- All semantic definitions are born equal
  - Private sector, cities/towns/municipalities, individual public sector ICT actors can proceed even before the government’s perspective of ”common semantics” is established

- The dominating semantics that mainstream will adopt is established with natural evolution
  - Integrations/migrations between models of different semantics is straightforward non-technical
  - Government chosen common semantics can be challenged (this is a good thing!) and so it can and will also evolve through natural development

... thus from Finnish entrepreneur's perspective doing European business...



Finland vs. Estonia?

- Private businesses within EU are all on domestic market
  - The businesses report to their countries - internally
- EU-Citizen's perspective is the whole EU
  - ... And as internet-enabled consumer, USA as well...
- No reason to over-underline nationality in global services
  - Especially on aspects that allow global leadership if treated right

To Compete or Co-Operate?

- ”The Ball” core team co-operates with everyone
- We just unintentionally ”compete” with parties who don’t for some reason yet co-operate with us ☺
- In the end our competition can still use our open platform to ”compete against” our open offering. Does that count as co-operation or competition?