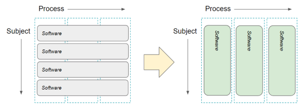
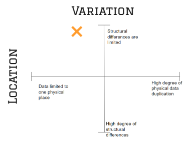
Business drivers - "Why?"

From subject matter oriented to shared process oriented software



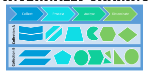
Enterprise data management

- Metadata
- Data lake
- new data sources
- Harmonizing data

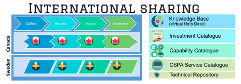

Variation
Location
Structural differences are limited
Data limited to one physical place
High degree of physical data duplication
High degree of structural differences

Reuse and sharing services

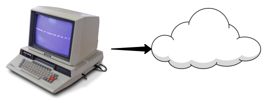
internally sharing



International sharing



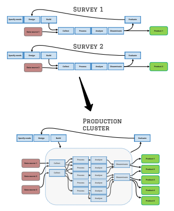
From legacy technology to SOA and/or cloud



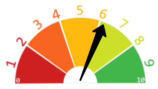
Agile business processes

Adapt quicker to new possibilities and threats

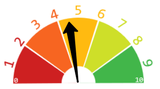
- New Data sources
- New "products" based on existing data
- Harmonizing statistics
- Quickly respond to new requests



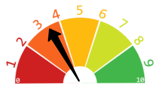
Specific advancements



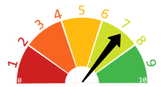
Security



User experience

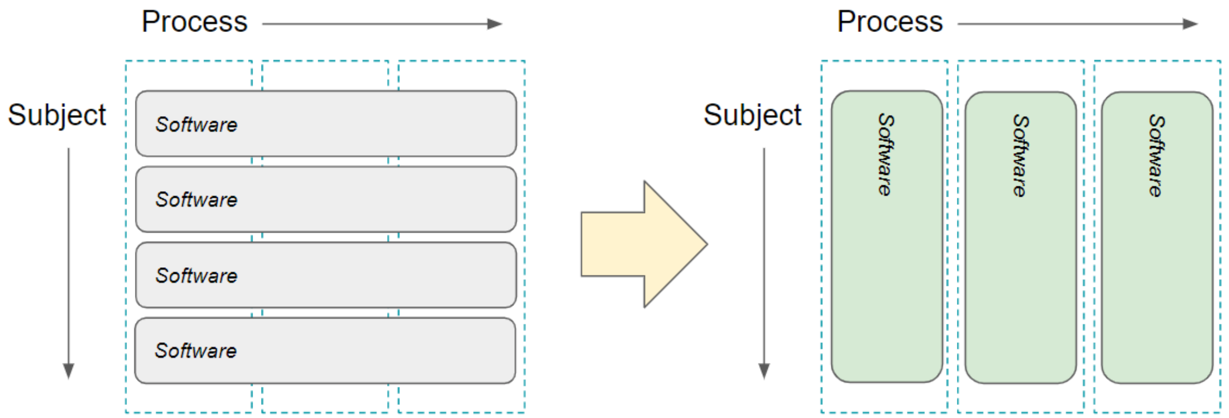


Integration



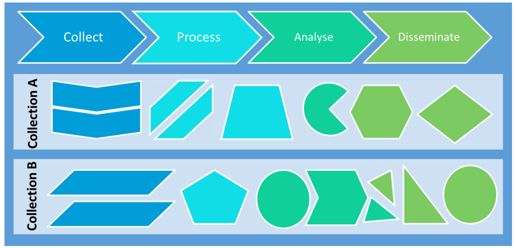
Open Data

From subject matter oriented to shared process oriented software

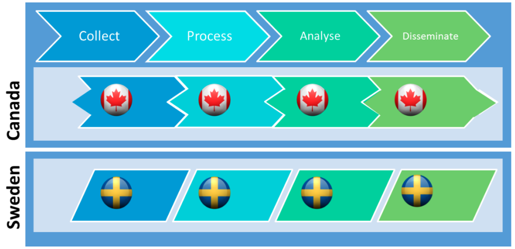


Reuse and sharing services

Internally sharing



International sharing

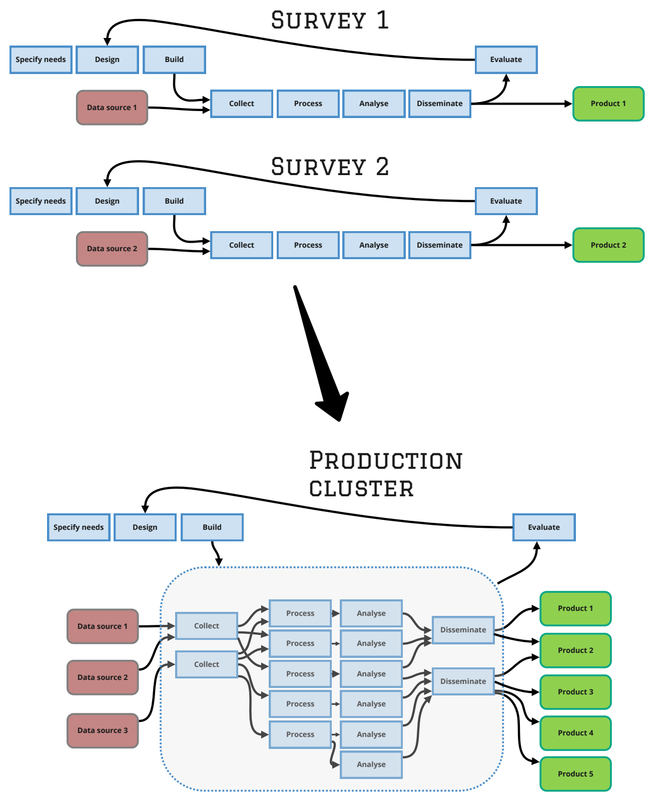


- Knowledge Base (Virtual Help Desk)
- Investment Catalogue
- Capability Catalogue
- CSPA Service Catalogue
- Technical Repository

Agile business processes

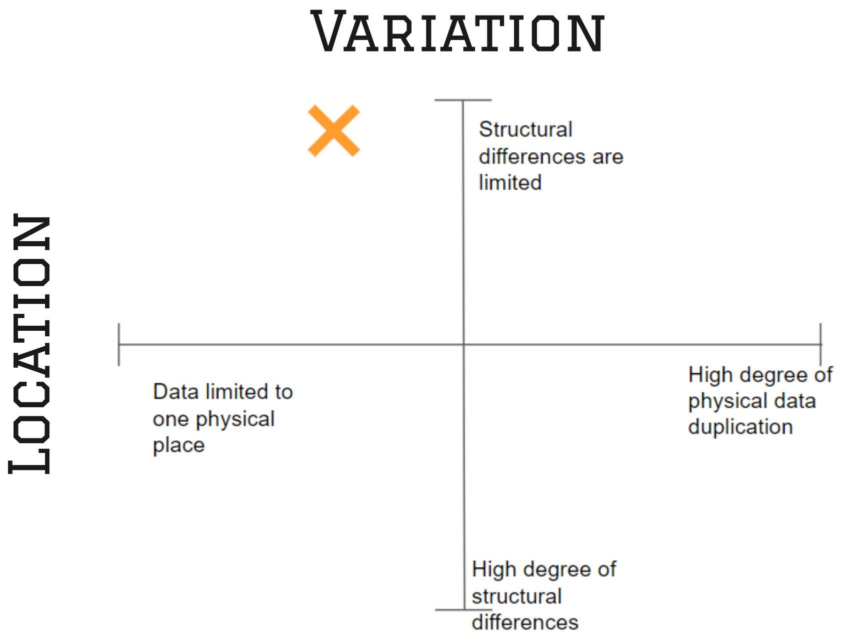
Adapt quicker to new possibilities and threats

- New Data sources
- New "products" based on existing data
- Harmonizing statistics
- Quickly respond to new requests

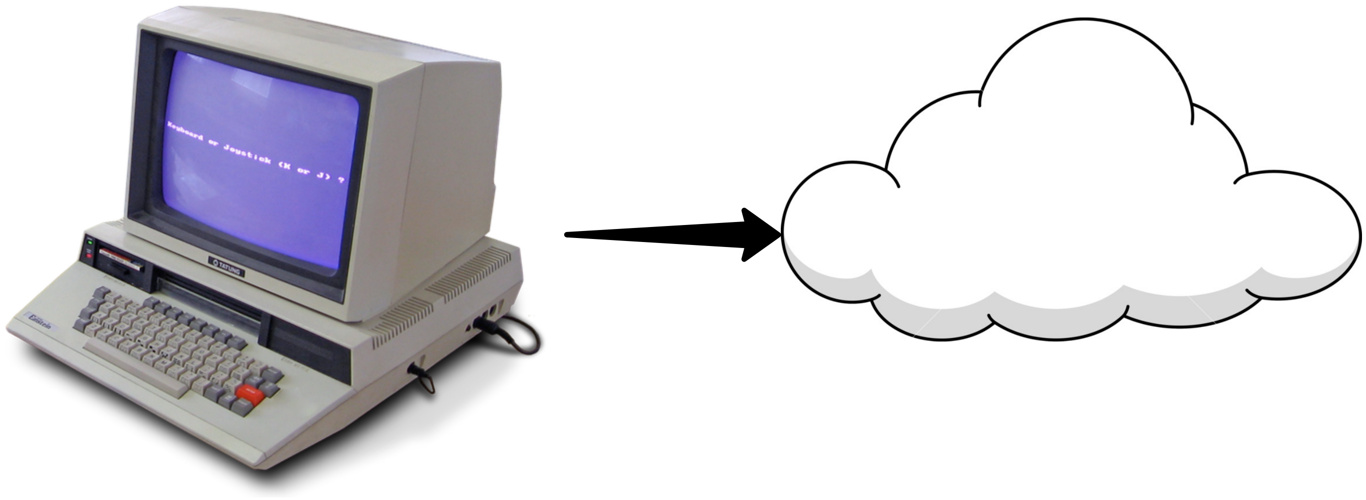


Enterprise data management

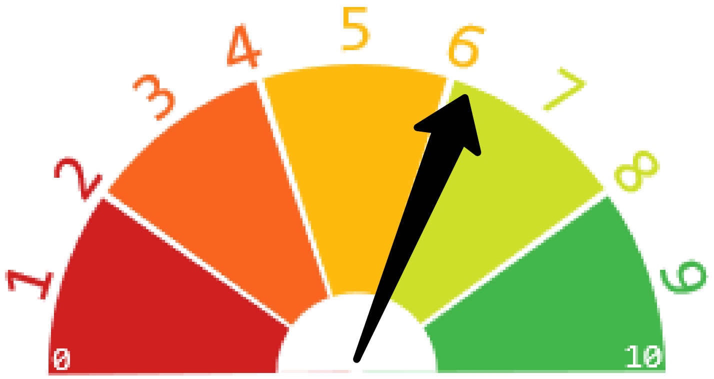
- Metadata
- Data lake
- new data sources
- Harmonizing data


Variation
Location
Structural differences are limited
Data limited to one physical place
High degree of physical data duplication
High degree of structural differences

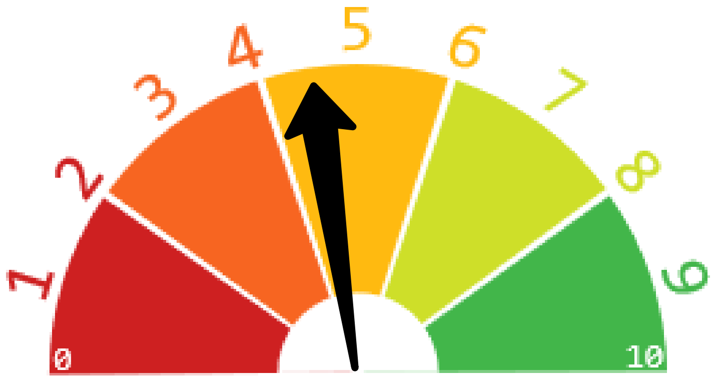
From legacy technology to SOA and/or cloud



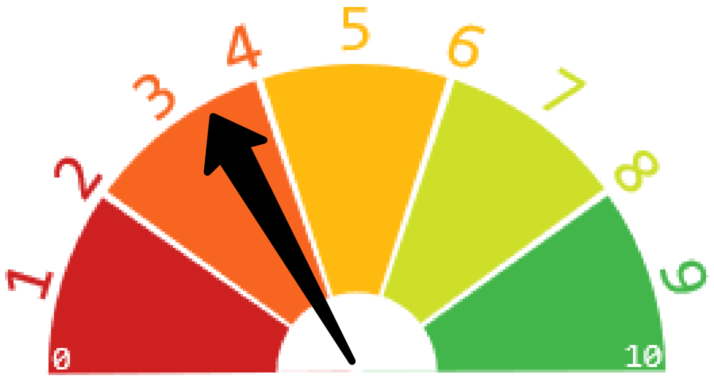
Specific advancements



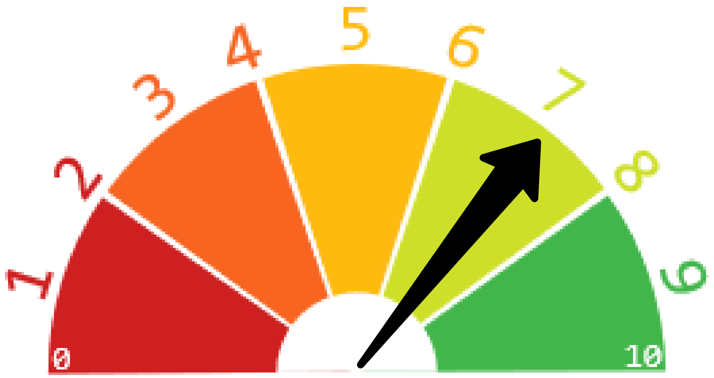
Security



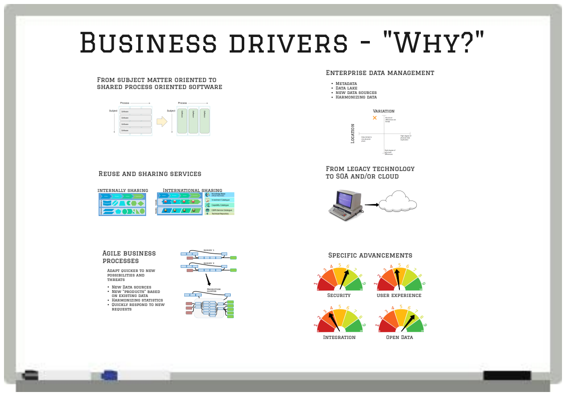
user experience



Integration



Open Data



Business drivers - "Why?"

From subject matter oriented to shared process oriented software

Enterprise data management

- Metadata
- Data lake
- new data sources
- Harmonizing data

Reuse and sharing services

From legacy technology to SOA and/or cloud

Agile business processes

Adapt quicker to new possibilities and threats

- New Data sources
- New "products" based on existing data
- Harmonizing statistics
- Quickly respond to new requests

Specific advancements

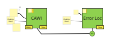
I3S-ESSnet/ExampleServices (github.com)

Scenario 1

Starting point

We start with two services (CAWI, ErrorLocalization). CAWI is a web based tool for taking surveys. ErrorLocalization is a service that can flag answers as faulty based on a rule set. The services or application do not know of each other, each has implemented its own implementation of a code list. Therefor each service need logic for handling of the codes lists used.

Work Item 1 - create the two services from the current code



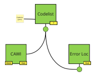
DEMO - starting point
Show how to start-up CAWI and ErrorLoc. Each services is started with CodeList.json as parameter.
Start Web app for CAWI and show Code List in drop down (read from CodeList.json)
Call ErrorLoc service from Dev-environment or CAWI - get and show response
Change CodeList.json for Error Loc (remove GB) and restart service - the result is not present in the other service.

Goal

Work Item 2 - create Codelist service and make changes to CAWI and ErrorLoc to use the REST service Codelist

[illegible]

Thought process
By extracting the handling of metadata to separate service we create a single point of thruth regarding data consistency.



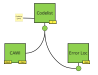
DEMO - Goal
Show how to start-up CAWI, Codelist and ErrorLoc. Codelist is started with CodeList.json as a parameter.
Start Web app for CAWI and show Code List in drop down (read from CodeList.json)
Call ErrorLoc service from Dev-environment or CAWI - get and show response
Change CodeList.json and restart services - new codelist is present for all services

Having a disconnected state of applications the manual labour to keep data up to date is high. The risk of poor data integrity is high. By extracting logic of manupilating the codelist metadata we have lowered the cost of keeping data up to date. We have now a masterdata repository who is owner of its data.

Scenario 2

Starting point

We have the three services CAWI, CodeList, and ErrorLoc but we have identified they use the same meta information in some regard. The dupication of information is therefore a fact. By making the services context aware we can extract logic regarding setting up duplicate meta information in the applications.

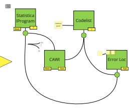


DEMO - starting point
Start a CAWI service with the EU Weather Survey. Start CAWI and ErrorLoc.
Try to start a second CAWI with EU Food Survey - a new service instance must be started instead of re-using the current service.
Same problem for Error Loc.

Goal

Work Item 3 - Create Statistical Program service. Make changes to CAWI, Codelist, Error Loc to be able to handle multiple contexts

Thought process By



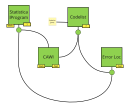
DEMO - Goal
- The user can change context between different statistical programs.
- The user can change between different serveys.

By extracting a service for handling meta data regarding Statistical programs we can minimize the duplication of manual setup within each service.

Scenario 3

Starting point

The services are now context aware. But we now want to look into the feature service integration and implement another pattern for integration. In this case an eventdriven patterna with publish & subscribe.

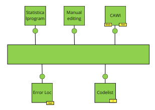


DEMO - starting point
The integration pattern between the services is Point-Point.

Goal

Work Item 4 - Establish a pub/sub environment, update services to make use of pub/sub.

Work Item 5 - Add the new service Manual editing - show that changes don't have to be made to existing services



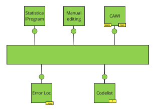
DEMO - Goal
The integration pattern between at least two services is changed to pubsub.

Within statistical production, multiple patterns will be used to solve different problems, for example, an event-driven approach may be suitable for metadata service integrations while point-to-point may be suitable for integrations where large datasets are exchanged.

Scenario 4

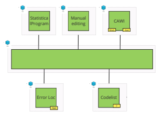
Starting point

In this scenario we want to implement the containerization feature.



DEMO - starting point
The services runs nativly on the computer.

Goal



DEMO - Goal
The services runs in av virtual environment in a docker container.

Dependencies on underlying technology and infrastructure are eliminated which increases the number of hosting options for service consumers. This allows for cloud hosting as well as flexible on-premise infrastructure alternatives

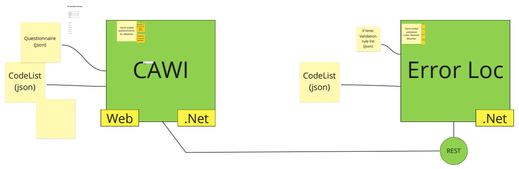
[I3S-ESSnet/ExampleServices (github.com)]

Scenario 1

Starting point

We start with two services (CAWI, ErrorLocalization). CAWI is a web based tool for taking surveys. ErrorLocalization is a service that can flag answers as faulty based on a rule set. The services or application do not know of each other, each has implemented its own implementation of a code list. Therefor each service need logic for handling of the codes lists used.

Work Item 1 - create the two services from the current code



Questionnaire (json)
CodeList (json)
Hard-coded questionnaire: EU Weather
CAWI
Web
.Net
If time: Validation rule list (json)
CodeList (json)
Hard-coded validation rules: Realistic Weather
Error Loc
.Net
REST

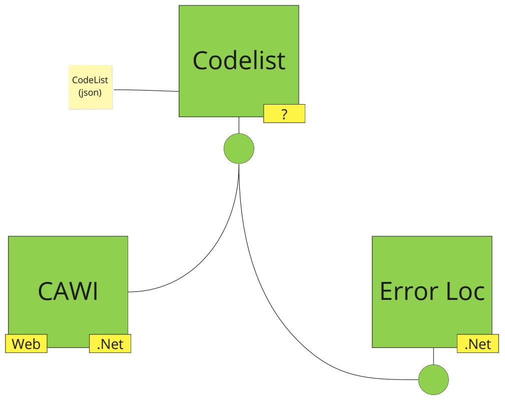
DEMO - starting point
Show how to start-up CAWI and ErrorLoc. Each services is started with CodeList.json as parameter.
Start Web app for CAWI and show Code List in drop down (read from CodeList.json)
Call ErrorLoc service from Dev-environment or CAWI - get and show response
Change CodeList.json for Error Loc (remove GB) and restart service - the result is not present in the other service.

Goal

Work Item 2 - create Codelist service and make changes to CAWI and ErrorLoc to use the REST service Codelist

To follow principle of metadata driven systems, we see that each of the two services needs to manage the same meta data. By extracting this function we can make each service be more accomodated towards single responsibility. Thus lowering the manual labour keeping two systems up to date with the same data.
S1

Thought process
By extracting the handling of metadata to separate service we create a single point of thruth regarding data consistency.



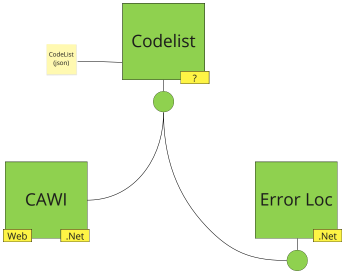
DEMO - Goal
Show how to start-up CAWI, Codelist and ErrorLoc. Codelist is started with CodeList.json as a parameter.
Start Web app for CAWI and show Code List in drop down (read from CodeList.json)
Call ErrorLoc service from Dev-environment or CAWI - get and show response
Change CodeList.json and restart services - new codelist is present for all services

Having a disconnected state of applications the manual labour to keep data up to date is high. The risk of poor data integrity is high. By extracting logic of manupilating the codelist metadata we have lowered the cost of keeping data up to date. We have now a masterdata repository who is owner of its data.

Scenario 2

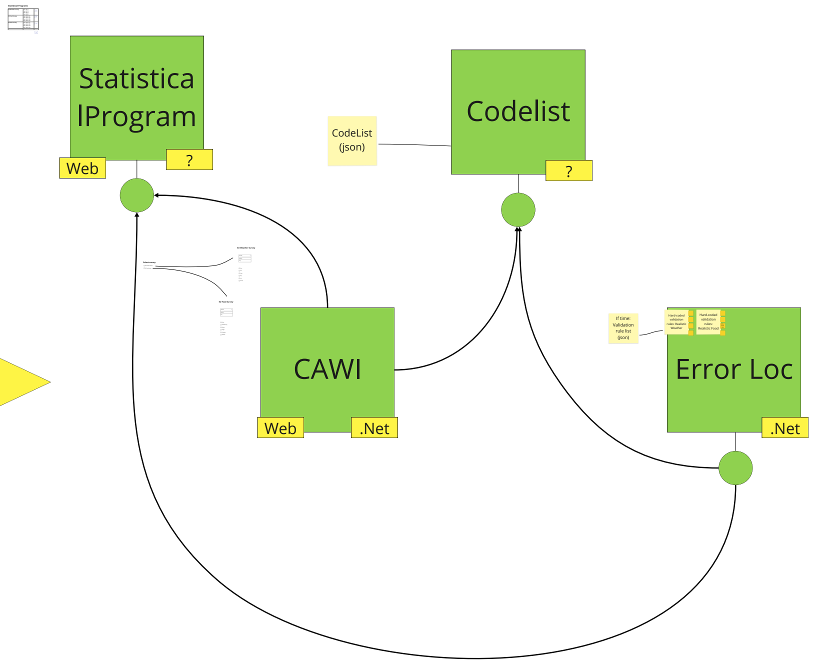
Starting point

We have the three services CAWI, CodeList, and ErrorLoc but we have identified they use the same meta information in some regard. The dupication of information is therefore a fact. By making the services context aware we can extract logic regarding setting up duplicate meta information in the applications.



DEMO - starting point
Start a CAWI service with the EU Weather Survey. Start CAWI and ErrorLoc.
Try to start a second CAWI with EU Food Survey - a new service instance must be started instead of re-using the current service.
Same problem for Error Loc.

Goal



Work Item 3 - Create Statistical Program
vice. Make changes to CAWI, Codelist, Error
oc to be able to handle multiple contexts

Thought process By

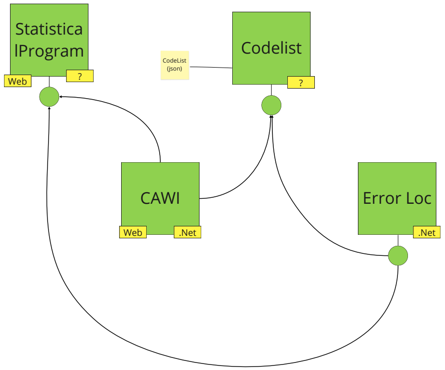
DEMO - Goal
- The user can change context between different statistical programs.
- The user can change between different serveys.

By extracting a service for handling meta data regarding Statistical programs we can minimize the duplication of manual setup within each service.

Scenario 3

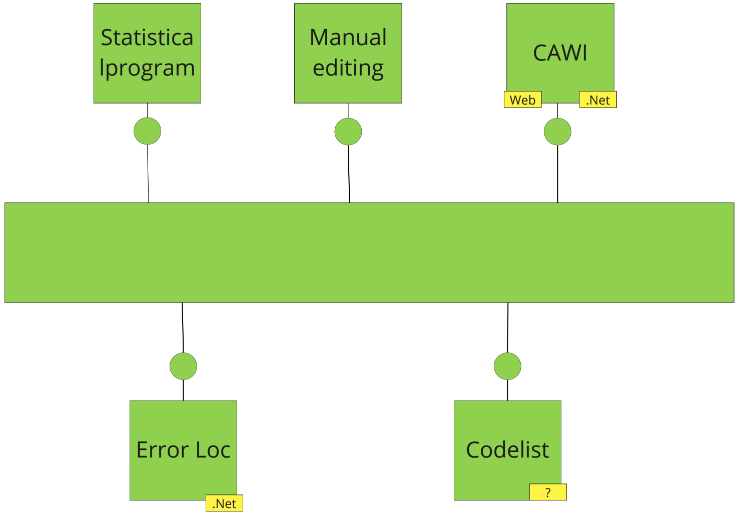
Starting point

The services are now context aware. But we now want to look into the feature service integration and implement another pattern for integration. In this case an eventdriven patterna with publish & subscribe.



DEMO - starting point
The integration pattern between the services is Point-Point.

Goal



Work Item 4 - Establish a pub/sub environment, update services to make use of pub/sub.

Work Item 5 - Add the new service Manual editing - show that changes don't have to be made to existing services

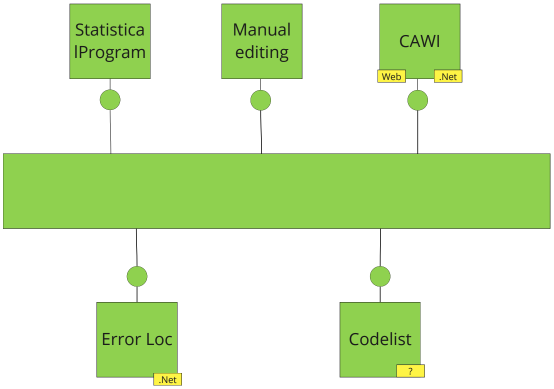
DEMO - Goal
The integration pattern between at least two services is changed to pubsub.

Within statistical production, multiple patterns will be used to solve different problems, for example, an event-driven approach may be suitable for metadata service integrations while point-to-point may be suitable for integrations where large datasets are exchanged.

Scenario 4

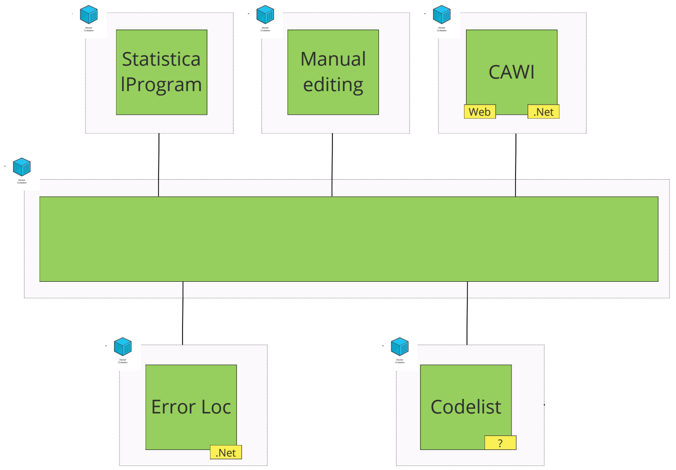
Starting point

In this scenario we want to implement the containerization feature.



DEMO - starting point
The services runs nativly on the computer.

Goal



DEMO - Goal
The services runs in av virtual environment in a docker container.

Dependencies on underlying technology and infrastructure are eliminated which increases the number of hosting options for service consumers. This allows for cloud hosting as well as flexible on-premise infrastructure alternatives