SOURCE™

Software Outreach and Redefinition to Collect E-data Through MOTUS

Implementation of a Governance Model to use MOTUS

Methodological and evaluation report Eurostat
Grant number: 847218 — BE-2018-TUS
Belgium, April 2020

Contact information

Project coordinator – Statistics Belgium

Kelly Sabbe

Statbel, the Belgian statistical office

King Albert-II-street 16

1000 Brussels

+32 (0)2 277 66 30

kelly.sabbe@economie.fgov.be

Beneficiary – Destatis (Statistisches Bundesamt) (DE)

Elke Nagel, statistician

Gustav-Stresemann-Ring 11

65189 Wiesbaden

+49 (0)611 75 8572

elke.nagel@destatis.de

Birgit Lenuweit, senior statistician

Gustav-Stresemann-Ring 11

65189 Wiesbaden

+49 (0)611 75 8572

birgit.lenuweit@destatis.de

Subcontractor – MOTUS developer

Joeri Minnen

hbits CV – Spin Off Vrije Universiteit Brussel

Witte Patersstraat 4

1040 Etterbeek

Tel: +32 497 18 95 03

Joeri.Minnen@hbits.io / Joeri.Minnen@vub.be

CONTENTS

List of tables

List of figures

(hbits.7) Implementation of a governance model to use MOTUS

Deliverable	The focus within this subtask lies within the necessity to provide multiple countries the opportunity to use MOTUS. Therefore a governance model needs to be defined to give national institutions a private entry to the MOTUS-software environment.

1.1 Definition of governance

In general terms governance refers to a process of governing where there is interaction and decision-making between stakeholders that have a common goal in the creation and usage of a product or a service. When the involvement of all parties is sufficient available, the decisions are acquired legitimate, and a defined path for the future can be designed.

Within the scope of this document three types of governance can be distinguished: Development Governance, Technical Governance and User Governance.

1.2 Development Governance

This part relies on the 'Governance Guide' presented to the Joint Task Force TUS & HBS held in November 2019 in Luxemburg. Below is a summary, the full text is available online[1]. In this document there is a high correlation between what is meant by 'Development Governance' and 'Governance Guide'.

The development governance deals with a framework that supports decision-making and advice about future needs, maintenance and sharing of tools and sources on ESS-level, both for TUS and HBS. The members of the Task Forces TUS and HBS in the first place, and second the members of the Working Groups of TUS and HBS have constituted this framework into a Governance Guide.

Members States showed their appreciation to receive guidance for situations where tools are available for sharing. The guide supports Member States to explore and test the tools, but also in their trajectory to implement the solution in their own data collection approach.

The implementation trajectory may also lead to a further development request to the System owner of the data collection tool based on the feedback after using or testing of the solution. Based on this information also the TFs and Eurostat obtain a better view on new releases and the estimated budget for new functionalities.

The guide defines the procedures to be followed in different cases, like looking for a new solution or having a need for new functionalities. The guidelines presents the cases as different situations. In every situation there is a contact between the NSI and the System Owner, while Eurostat is being kept into the loop.

In the document the System Owner is assigned three tasks:

- Take care of country questions while the tool should also stay shareable on ESS level
- To be responsible and maintain the tool, module or plugin
- To update the CSPA-information in the Tools & Source Inventory[2]

Examples can be discussions about the logo/template management, and the languages being available in the application, and how to make these options available to the Member States.

The software platform MOTUS is able to handle according the philosophy of the Governance Guide.

The Development Governance deals with new and missing functionalities. The next part discusses the more technical implementation of a software tool to become shareable in respect to visions expressed in the Governance Guide.

[1] https://webgate.ec.europa.eu/fpfis/wikis/display/ISTLCS/2019+November?preview=/393220501/413639884/TF_JOINT_Innovation_201911_2.1_Governance%20Guide.%20V1.0.pdf

[2] The new inventory (V4.0): https://webgate.ec.europa.eu/fpfis/wikis/display/ISTLCS/INVENTORY

1.3 Technical Governance

MOTUS has not only a front-office (web and mobile) which shows the functionalities towards the respondent, but also an extensive back-office that is available to the researchers. The software architecture is been documented in HBITS.2. The communication between the components is being taken care of through different APIs.

At the same time MOTUS aspires to a 3-tier application architecture, see HBITS.6. A 3-tier application architecture is a modular client-server architecture that consists of 3 parts: (1) the client tier with the GUI code; (2) application tier with the business logic; and (3) the data tier with the stored information.

All these elements support the ethical, privacy and security setup of MOTUS and which should be requirements for a future ESS platform that is also shareable. These values should be part of a sound audit of the MOTUS-code (available in a git repository for version management) to document the achievements, but also to highlight (probably) necessary developments on the functional, technical and security level to let MOTUS unfold itself to an ESS-shareable platform.

Independent of this audit, the most important question is how to govern the MOTUS-code, so that the outcome of the code is available to the NSIs while at the same time comparability in the data collecting is guaranteed. The answer lies within the strategy to host MOTUS. In this document four variants are described. Other permutations are possible.

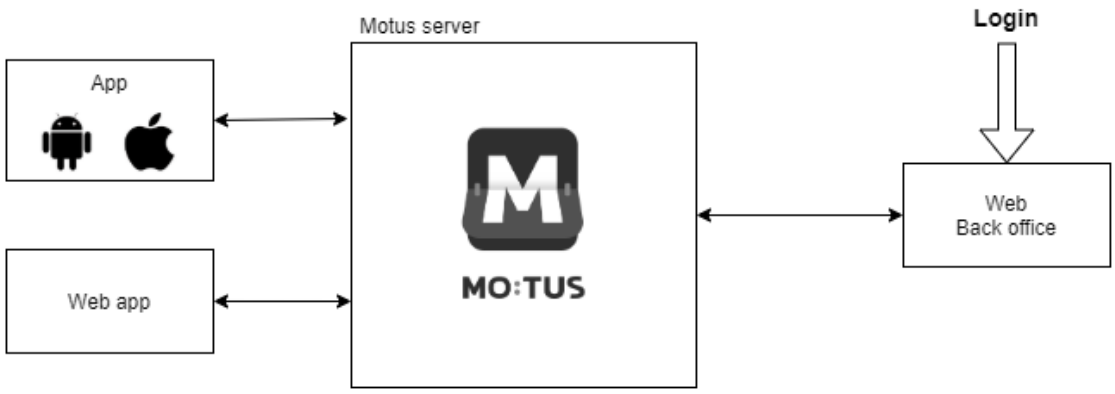
1.3.1 Architecture A: MOTUS as a service

In Architecture A all the technicalities of the hosting are handled by MOTUS. The NSIs can receive a private entrance to the MOTUS back-office to setup and run their private fieldwork (in line with the harmonization agreements). The data go and are stored in the MOTUS-datacenter. The datacenter exists outside the NSIs. MOTUS has access to all servers and unencrypted data.

The hosting is a shared hosting model. The servers handle multiple studies in parallel. The isolation of the data is granted on a logical layer.

The code is maintained by the MOTUS-team in accordance to the agreements made in the Government Guide.

Figure 1: Architecture A - MOTUS as a service

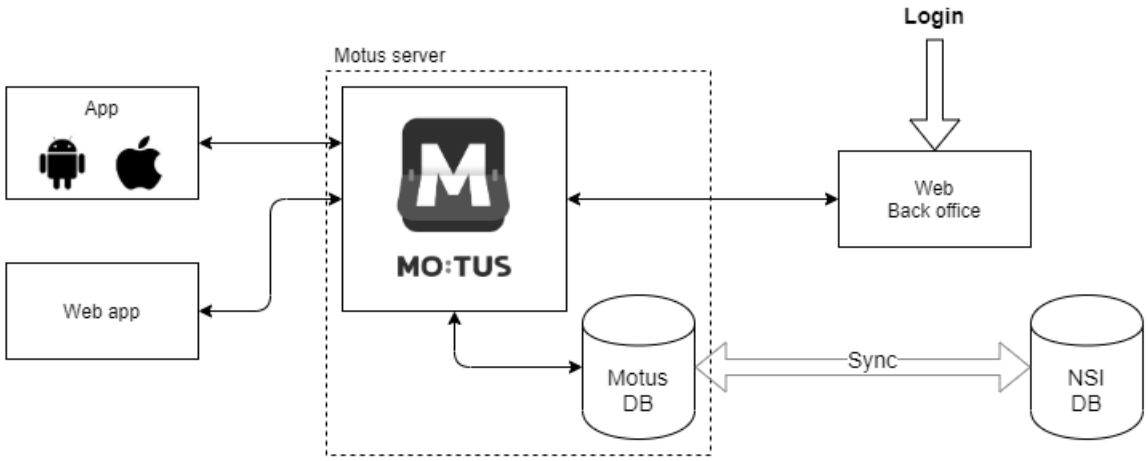


1.3.2 Architecture B: MOTUS as a data collector

Architecture B differs from Architecture A in the replication of the database to a database in the NSIs datacenter. The replicated database is fully available to the NSIs for further processing. E.g. to develop a service which sends the data to a processing environment.

This replication and development of extra services has as extra value that the NSIs keep the processing strategies private from MOTUS. A disadvantage is the unawareness of the NSIs about the MOTUS database schema, which the team behind MOTUS has. The code is just like in Architecture A maintained by the MOTUS-team.

Figure 2: Architecture B - MOTUS as a data collector



1.3.3 Architecture C: MOTUS virtualized

In Architecture C all servers are hosted by an NSI. Every NSI needs their own server capacity in their own datacenter. The server setup needs to meet some requirements in order to let MOTUS run properly. No data is available outside the datacenter of an NSI.

The MOTUS-code runs on the 'application tier' of the NSI. The code is containered via the use of the Docker technology. It runs on the private servers of a NSI, and the MOTUS-team keeps the full control over the MOTUS-code, which has advantages for development, maintenance and to support comparability between TUS and HBS studies across Europe, while at the same time shareability is guaranteed.

With Docker the MOTUS-application and its dependencies is turned into a package, or a so called Docker-image. This process is also called 'virtualization of software'. A virtual container can run on any Linux server, which is located at the host datacenter. Docker is one example of a software platform that use OS-level virtualization to deliver software in packages. Another option is Kubernetes. Both their focus lies on providing consistency across platforms form development to production. MOTUS has made the choice to work with Docker.

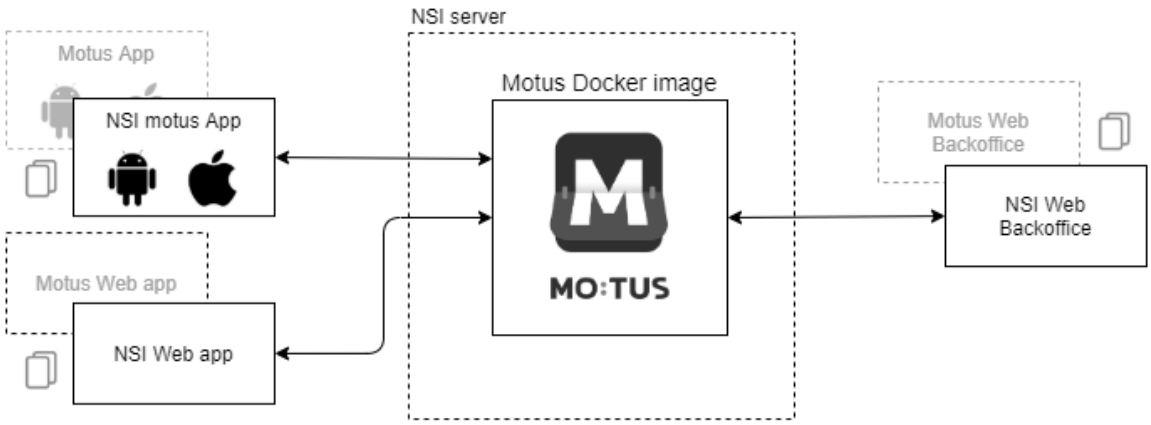
Multiple docker images can be developed, in numbers but also in content. On the one hand it means that a docker image is available to multiple NSIs with the same possibilities, while on the other hand variation(s) can be supported through Microservices (or Plugins) which can also be dockerized. It is also possible to develop different containers for individual components of an application. An example is the database. The different containers then can communicate with each other via the network. An extra

advantage is that developers from different organizations can share their applications, or components, to arrive to a joint application. This supports collaboration between different organizations, while respecting the specific needs of (e.g.) the Members States' specific environment.

Once a docker image is defined, it is easy to distribute. The images are available in a registry which is hosted in the MOTUS datacenter. Due to the Docker software even complex structures (with independent containers) can be linked to function as one application through the Docker-compose function. Through Docker, applications can be properly scaled since more server nodes can be filled with containers when the demand grows.

Architecture C provides the building blocks for the 'industrialization' of the MOTUS-software, and so the flexibility towards the NSIs to choose from the list of virtual containers and at the same time develop solutions of their own that can communicate with each other. In this perspective NSIs can exclusively handle their own surveys.

Figure 3: Architecture C - MOTUS virtualized

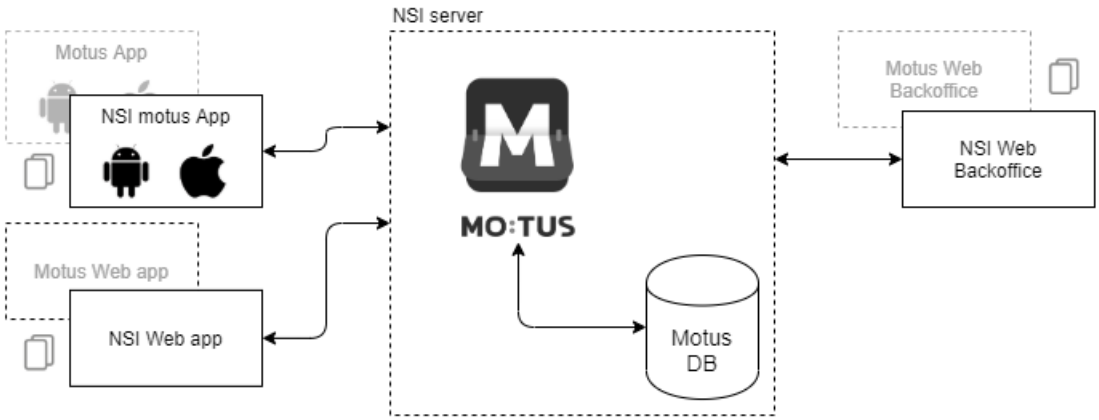


1.3.4 Architecture D: MOTUS native installation

In this last variant Architecture D a NSI hosts the whole MOTUS system is their datacenter, just like in Architecture C. Different from option C is that MOTUS is installed natively, and so a NSI has a complete insight into the source code and the server applications. MOTUS would receive a remote access via SSH to provide technical assistance.

With Architecture D it becomes likely that different versions of MOTUS will be developed and that comparability and shareability are jeopardized, as well as there is a breach towards to philosophy to establish an ESS-platform (see also the contrasts with the Governance Guide). Also, and in contrast to the other options the legal framework would become more prominent in the relationship between MOTUS and NSIs.

Figure 4: Architecture D - MOTUS native installation



1.3.5 Four different architectures: one conclusion

In the presentation of the four different architectures different criteria were discussed. To provide an overview a correspondence table is provided below. In total 21 criteria are scored with + and – with 3 different grades (+++/---). The grading is helpful to define which strategy is the most valid option within the scope of Technical Governance. In short it offers an architecture quality assessment.

Table 1: MOTUS governance correspondence table

Criteria	MOTUS-governance: options			
	A	B	C	D
	Full Service	Data collection	Virtual application	Native installation
Simplicity	+++	+++	--	---
Stability	++	+	+++	-
Relational cohesion	+++	-	+	--
Maturity	+++	+	++	---
Efficiency	+	+	+++	-
Maintainability	+++	-	+++	---
Responsibility	+	++	++	--
Support	+++	+	+++	--
Usability	--	--	++	+++
Suitability	+++	-	+	++
Extensibility	+++	+	+++	---
Scalability	++	+	+++	---
Interoperability	+++	--	+++	--
Availability	+++	+++	+	-
Security	+	-	++	+++
Comparability	--	-	+++	---
Country specific	--	+	+++	---
Shareability	+	-	+++	---
Cost installation	+++	+++	-	---
Cost update	+++	++	+	--
Legal	+	++	+++	---

Option A and C gain higher scores than B and D, where D is ranged by far the lowest. In favor of Architecture C plays the overall positive grades for all parties, while the negative grades are more linked to one-off costs. After the setup of the virtual docker container the advantages are available for all parties. A run-through within the Architecture C could be:

- Developers write code locally and share their work with other colleagues or developers in different organizations using Docker containers.
- Docker is used to push the applications into a test environment and to execute automated and manual tests.
- The same with bugs handling, bugs can be fixed straight-away in the development environment and redeployed to the test environment for testing and validation.
- When the test phase is complete the updated application is pushed to the production environment as a simple updated image.

An important value of Architecture C is the close and natural cooperation between people from different institutions, as well as within the same institute. This is partly the story of the next governance type: the User Governance.

1.4 User Governance

When the document discusses the User Governance, the User is considered to be a researcher or a group of researchers with shared responsibilities, tasks and restrictions. In this respect the User Governance has to deal with different clients. And within a client with different roles. In the first case it is called the multi-client capability, in the second case it is called role definitions.

1.4.1 Multi-client capability

A multi-client software is defined as a software architecture which has the source code running on one server, but which provides the opportunity for multiple clients to make use of it.

As being documented in HBITS.1 has MOTUS an extensive back-office, called the MOTUS-builder. The MOTUS-builder is built up by 4 phases: design, collect, analyze & advice (DCAA). Every phase has some underlying components, called builders. There are two options: a single instance and a multi instance.

1.4.1.1 Single instance

Today the MOTUS-builder runs on one server. From this installation different MOTUS-environments can be defined. This means that every NSI can have its own MOTUS-builder environment and no information is shared between different MOTUS-environments. Having entrance to an own builder means that all functionalities become available to an NSI.

A disadvantage is that the data is not fully separated in the database.

1.4.1.2 Multi instance

With the multi instance a separate instance is created for every client/NSI. This means every NSI has its own subdomain and its own MOTUS-builder environment. In this option the data is separated in different docker containers, so it is ensured that no data is leaked between different NSIs, and role management is being done on the level of each NSI separately. An extra advantage is that the load is spread of the different containers.

The multi instance setup shows how MOTUS can be employed by different Member States separately from each other. And without any influence and leakage of data. But also within a country specific setup this strategy can have its add value.

An example could be where different states or provinces have the responsibility to collect the data from the inhabitants in their state or province, but where the federal government has the overall responsibility and the data are stored centrally. In this example the NSI is responsible for the overall template and the overall structure of the study design, and at the end for the export, the validation and the processing of the data. The state or province is than responsible for the personalization of the invitation, the creation of logins, the fieldwork (support) and (non-)response follow-up. Also the export and validation belongs to their tasks.

Because the data rests within different subdomains and so different sub databases, a disadvantage is that an NSI has no global overview of performance of all the states or provinces. To support the global overview, one way to handle this disadvantage is to aggregate the data from multiple instances (i.e. multiple databases and so different id's for the same variables) of the states or provinces through the Analyze server via a specific script. In doing this an instantaneous global overview of the performance of all the states or provinces can be provided.

Data analysis and preparation can be further processed via the inclination into R (see HBITS.11), and visualizations could be used to give an insightful overview to a dashboard within the NSI MOTUS-builder.

Both the single and the multi instance can be further diversified through role management.

1.4.2 Role management

Role management is a control mechanism with defined roles and privileges. Roles are related to various job functions and these functions come with permissions to be able to perform the operation that are necessary to execute this role. The roles are defined by an organization and the administrator assigns persons to a role.

Functional roles could be:

- Administrator
- Team leader
- Staff
- Archivist

These function roles can be further specified on the technical level. Within a role also different accesses within the MOTUS-builder can be configured, without limitations. However a full administrator can never be blocked/changed. Technical aspects can also mean restricted entrance to one of the defined builders in MOTUS:

- Device builder
- Survey builder
- Diary builder
- Event builder
- Communication builder
- Language builder
- Research builder
- Invitation builder
- Dashboard builder
- Data builder
- Quality builder
- Computation builder
- Visualization builder

Both the functional and the technical role can overlap, as they can also have permissions to only read, to adapt, or to ask confirmation.

Today some roles are defined within MOTUS. Other roles need to be implemented but are very feasible. Once the roles are defined in MOTUS different roles can be combined.

1.4.3 License approach

A license approach is based on the choice of the architecture, and on the common goals of the different NSIs.

To date it is not yet feasible to come to a license model.

1.5 Conclusion

This report has a strong emphasis on how to distribute the MOTUS software platform to other NSIs and institutions. Therefore a governance model needs to be implemented.

Such a governance model has 3 levels: (1) a Development governance, (2) a Technical governance, and (3) a User governance. The first governance is on the table of the Joint Task Force TUS & HBS.

Most interesting is the evaluation of the different technical architectures to give access to MOTUS. Based on 21 criteria it shows that the installation of a virtual machine (Architecture C) is the most promising to arrive to a true ESS-platform that gives high values to shareability and comparability.

The report finishes with ideas on the user governance, and more in detail about the aspect of 'Multi-client capability' and 'Role management'. It is clear that developing a software platform is far more reaching than only having a good User Interface. Also the daily actions and roles within a NSI should be covered. Only then a platform can be stretched out over different phases of the GSBPM architecture.