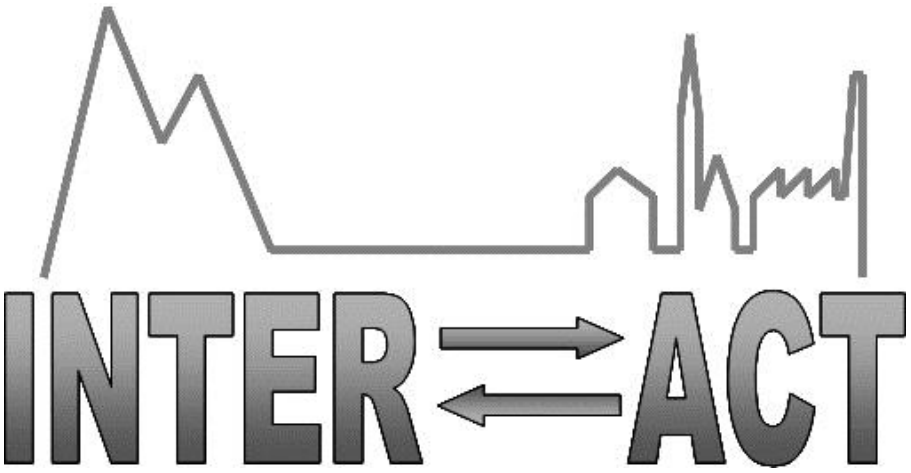
Combination of CP & CSA



D7.7- ScanDB Usability

Project No.262693– INTERACT

FP7-INFRASTRUCTURES-2010-1

Start date of project: 2011/01/01 Duration: 48 months
Due date of deliverable: 31/12/2013 (M36) Actual Submission date: 01/01/2014

Lead partner for deliverable: ITU
Authors: Philippe Bonnet, Javier Gonzalez, Niels Martin Schmidt, Thomas Bauditz Hansen

Dissemination Level		
PU	Public	X
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the Consortium (including the Commission Services)	
CO	Confidential, only for members of the Consortium (including the Commission Services)	

Table of Contents

Publishable Executive Summary

This report (i) describes the architecture of the new dataset management platform (GEM Database Manager) and of the workflow defined at Aarhus University for data validation/publication,(ii) illustrates how the scientists involved in a monitoring program benefit from the refined workflow and we discuss its usability. We explain how this architecture leveraged some of the design decisions from ScanDB.

1. Introduction

ScanDB is a tool to help scientists to manage time-series of data coming from different sources and with different properties. Scientists usually collect data from many different sources that may vary in location, source, weather conditions, etc. Each source is likely to have different properties such as air temperature, wind direction or snow depth. Typically, this data is gathered in spreadsheets and self-made databases which are then used to summarize and analyze it (see D7.1 for details).
Since there is no standard way describing how to work with scientific data, each scientist defines her/his own workflow. This makes it hard for many scientists to work together and share data. ScanDB V0 aimed to help scientists organize their data, allowing them to import data sets into a repository, annotating changes and keeping track of how data is updated with time.
The system has inspired Aarhus University to define a new data platform supporting a new type of workflow for data management and data publication. This workflow which is being deployed for the Greenland Ecosystem Monitoring (GEM) programs could serve as an example for the rest of the INTERACT community.
This report thus (i) describes the architecture of the new data management platform (GEM Database Manager) and of the workflow defined at Aarhus University for data validation/publication,(ii) illustrates how the scientists involved in a monitoring program benefit from the refined workflow and we discuss its usability.

2. GEM Database Workflow

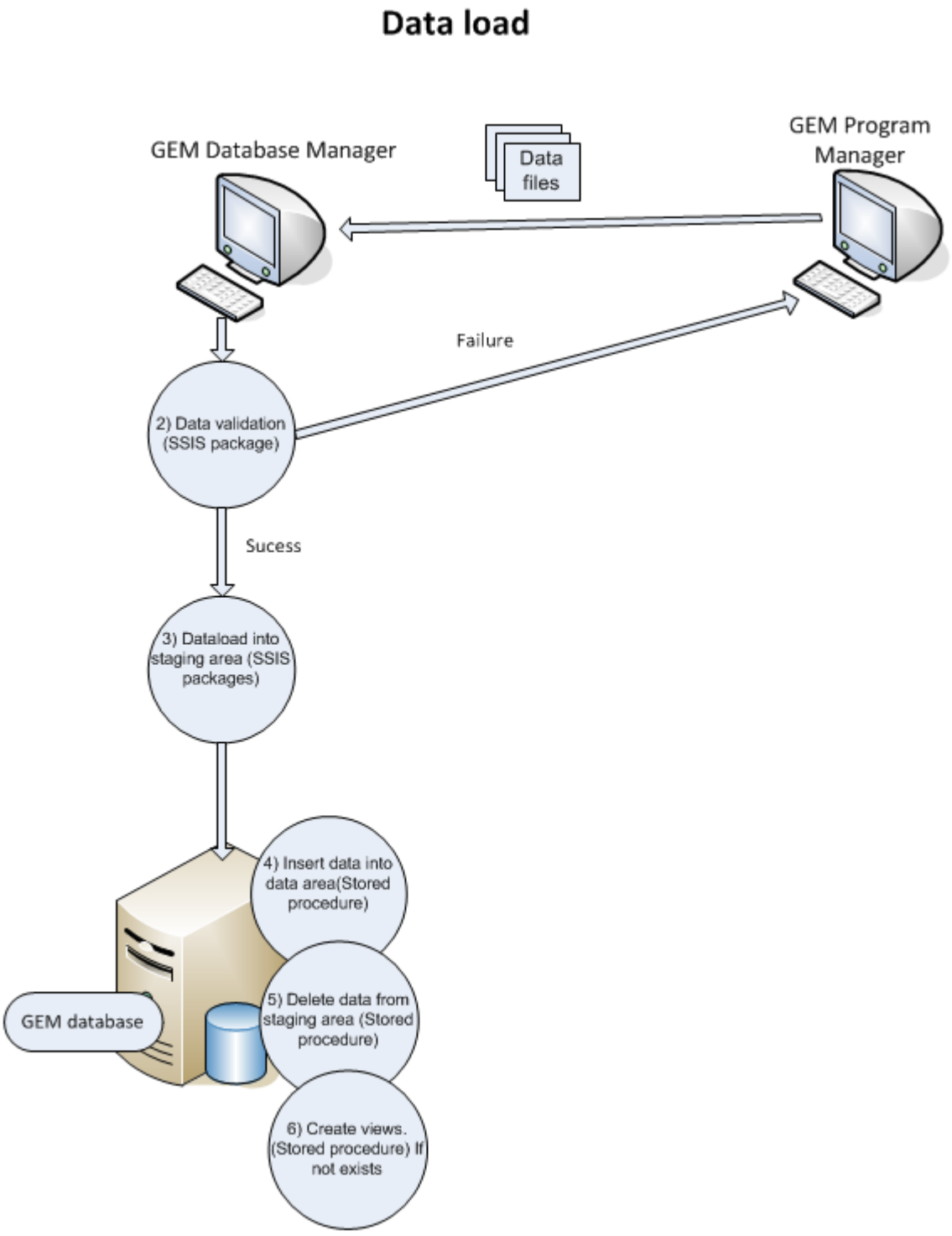


Diagram 1: GEM Data Loading Workflow

Diagram 1 illustrates the new data manager, the GEM database, based on SQL Server Integration Services SSIS and stored procedures. The workflow can be described as follows from the point of view both of the program manager and from the data manager:

- **The GEM Program Manager** centralizes the collection of data from the field (via sensors and personal devices used by technicians in the field) and proceeds to data validation/data cleaning with a combination of familiar tools (e.g., Excel) and historical data stored in the GEM database.
  It was the original goal of ScanDB to enable the GEM program manager to use historical data and advanced functionalities (lineage) during data validation. The GEM database now supports the use of historical data during this data validation phase. This is significant support for GEM program managers that can verify how trends and outliers observed in a batch of new data compares to historical data. It turned out that the use of familiar tools weighted more than the access to advanced functionalities such as lineage. As a result, ScanDB is not used as such in production. However, the principles of ScanDB are integrated into the GEM database workflow.
- **The GEM Database Manager** is in charge of data publication in the GEM database. He/she receives the data files from the GEP program managers and runs them through a data validation suite (implemented as a SQÆ Server Integration Service). In case data does not match the requirements of the underlying database, data validation fails and portions of the original data files are sent back to the responsible GEM program manager, whose responsibility it is to fix the detected problems.
  In a second phase, data is loaded into a staging area of the GEM database (again via a SSIS package). The data in the staging area is loaded in the format provided by the GEM program managers (described in the next Section). The data is then transferred from the staging area to the actual data area from the GEM database, additional views are created in the data area if necessary (via atored procedures).

3. Usability

3.1. Data Model

The GEM database is designed with flexibility in mind. The program manager works with data, in a row-oriented format. Different row formats might be required for various measurements or time series. For example, in the Zackenberg BioBasis program, data about Muskox is based on 26 attributes, data about *Eriophorum* (cotton grass) phenology requires 11 attributes, and anthropod emergence requires 40 attributes.

This creates a potential usability problem. Indeed, one option is to create a very complex database with tables that correspond to the manipulated data. This way data upload is straightforward. The problem is that a table needs to be created for each dataset, which requires unnecessary communication between program manager and database manager. Another option is to create a simple database, which constrains the way a program manager actually works, and negatively impacts the usability of the database.

To solve this problem, Aarhus University chose a solution similar to the one proposed in ScanDB – see Appendix A in Deliverable D7.3) based on the following features:

1. The following generic schema is defined:

(Program Element, ElementGroup, Location, Element, ColumnName, Unit, DataType,Field Description)

2. Using this generic schema, the program manager defines for each data set (denoted as an element in the context of a program element/element group, and at a given location), a collection of attributes defined with a column name, a measurement unit, a data type and a textual description.
   For example, the following attributes are defined for the *Eriophorum* total count, whose "program element" is BioBasis Zackenberg, "element group" is Vegetation, "location" is Permanent plots,:

Obsdate	Date	Date	Date of census
Observer	Text	Text	Name of field observer(s)
Plot	Text	Text	Plot ID
Section	Text	Text	Plot section ID
E_s_normal	Number	Number	Number of fertile *Eriophorum scheuchzeri* inflorescences
E_s_infertile	Number	Number	Number of infertile *Eriophorum scheuchzeri* inflorescences
E_t_normal	Number	Number	Number of fertile *Eriophorum triste* inflorescences
E_t_infertile	Number	Number	Number of infertile *Eriophorum triste* inflorescences
TotalCount	Text	Text	TOTALCOUNT indicates total inflorescences production in section
Field_remarks	Text	Text	Remarks from the field observer
General_remarks	Text	Text	General remarks to the census

   This way, each dataset is precisely defined in a way that gives (a) the program manager the flexibility to define the most appropriate representation for the data, and (b) the database manager a well-defined stable representation across data sets.
3. When the program manager submits a collection of data sets (each identified by a unique name – based on the concatenation of **Program Element, ElementGroup, Location, Element**), then the data can be validated using the attribute definition previously given (e.g., type checks can be enforced), transformed from the row format from the staged data into the generic format from the data area, and views can be defined to export datasets in the row format which is best suited for program managers.

3.2. Processes

Program managers manipulate data in the row-format obtained during data collection that has been used for years. Data is available to program managers in row format via custom views. This way, program managers can keep on using their favourite tools, possibly expanded with database access to leverage the historical data stored in the GEM database.
The generic format is only visible to the GEM database manager, who can define indexes, clustering and sharding policies to improve performance. Such database administration activities have no impact on usability from the program manager point of view.

4. Conclusion

ScanDB was a research prototype whose objective was to make life easier and more efficient for program managers. The central design decisions that underlie the GEM database are similar to the ones adopted by ScanDB : (i) a generic column-based database design exposed in row format to program managers, and (ii) data validation feedback loops between database and program manager. Such a design is a good example for dataset management in all INTERACT monitoring programs.