PROJECT PROPOSAL: Development and Implementation of a Fishery Data Management System

JS

January 12, 2013

Contents

1 Introduction 1

2 Background 1

2.1 General Context 2

3 Project Objectives 28

3.1 Data Input 28

3.2 Data Validation 30

3.3 Data Transformation 31

3.4 Data Output 33

4 Key Aspects of the System 41

4.1 AGILE Approach 41

4.2 Alternatives to the Relational Model 45

4.3 Redefining the Centre of the Application 47

4.4 Identifying Reuse Opportunity for Use Cases 48

4.5 Use of Free and Open Source Software 48

5 Aknoweledgments 52

References 53

1 Introduction

The project's longer-term development objective is to contribute to the sustainable management of the fisheries exploiting the living resources of the Ocean in general, in accordance with the Code of Conduct for Sustainable Fisheries and the Ecosystem Approach to Fisheries. The immediate objective is to improve the technological (software/hardware) component of the small-scale fisheries monitoring system. For that purpose, we are going to review the existent fishery data management systems, analyzing their strengths and their limitations, and propose a set of solutions, based on the latest technologies on databases and Geographical Information Systems. Finally, we will describe some guidelines about how-to implement this system, designed to overcome "traditional" obstacles faced by software development projects.

2 Background

According to [22], "decision making for fisheries policy-making, planning and management relies largely on processed information, not raw data". Often the datasets are so large, that the only way to deal with them is through a Database Management System (DBMS). The DBMS in this context may have many functions, but the most relevant ones are: to store the data and ensure its validity and compatibility; it should also be capable of processing it and outputting it, in different ways. Often the database functionality with its proprietary language, is not enough to cover the range of operations that we want to do with the data, and thus it needs to be extended by using a programming language external to the database.

When designing a complete system, many components should be take into account; namely, the User Interface (UI), the data entry, the data processing, the reporting and the Geographic Information Systems (GIS) functionality (that can be involved in many of the previous steps); the documentation is also something to consider, as it ensures the usability of the system, and some of the reviewed systems also tackle the component of knowledge transfer, through tutorials and workshops. It was not found any system that implemented all of these components, but they seem to focus on some of them instead (for instance, data processing). As long as the system focus on interoperability, by using open standards and free and open source software (see section 4.5), this should not be a problem since we can combine different systems for our purposes, and avoid in this way, the redundancy on development.

In terms of architecture, there are many options available for building such system; ranging from using complete commercial solutions, by tying together different components, to creating a *tailor-made* application; all of them have advantages and inconvenients. In this review, we tried to focus mostly on systems that are already in use, rather than in proposals or assessments of "how" systems should be. This is because to be put in place, fisheries management systems may face various obstacles, such as: burocracy, costs, etc; their ability to cope with them is also part of their success.

2.1 General Context

In this section we describe the results of reviewing 25 fisheries data management systems. The criteria for selecting the systems that integrate this review, was their visibility; either because they were referred by a colleague, or because they appeared in the internet - as a result of search queries, or within the site of some well-known institution/organization.

These were some key aspects that were taken into account, when reviewing those systems:

- security
- engaging and involvement of stakeholders
- validation
- ability to export information
- modular architecture and scalability
- licensing
- ability to work with incomplete data
- status (completed, draft, etc)

Whenever possible the software was tested, but in some cases, where this option was not available, the review is based on user manuals/articles or webpages. Finally it is important to note that often, the systems do not present any information about the type of license, even if sometimes they are distributed at no charge.

The results of this review, are summarized in a table, on a SQLite database [23]. In the next few paragraphs, we are going to analyze the content of this table, reviewing specific characteristics of these systems.

Most of the systems analyzed include some sort of database system, to store and manage information (see Figure 1a). The DBMS found are very diverse, ranging from free systems such as MySQL, to expensive proprietary solutions such as Oracle or MS SQL Server. Nevertheless, the majority of the systems use MS Access DB, which although commercial, is low-cost solution (and there are even some free versions).

The same conclusion can be drawn about GIS (see Figure 1b). The majority of systems uses a GIS component, or has a direct connection with GIS. However, the number of systems without GIS is close to the number of systems with GIS.

As we can see in see figure 2a, the vast majority of the reviewed systems does not require an internet connection, in order to work. Also because some of the systems, are low in technology and make little (or no) use of the computer. However, it is important to note, that in modern systems there is an increase use of the server-client architecture, where they centralize the information (database, maps), and serve it through "thin" clients (normally web browsers). The use of web-mapping services such as WMS, triggered a "boom" in the adoption of this kind of model. It is important to note, that although the big advantage of this model is its distributed use, it may also possible to setup it locally on a LAN, or even on a single computer,

Most of the reviewed systems, whether they use software or not, do not present any information about the type of license in place (if any). This can be a consequence of the fact that most of the systems are not distributed, since they are nowhere available for download.

Moreover, most systems do not require a high knowledge of GIS, databases, statistics, etc, in order to be used (see figure 2b). This is either because: costum made solutions were implemented (”tailor-made” software), or widely-known software is used (like for instance MS Access). However, in some cases, specific tools are used that require a bit more literacy (for instance R); in this cases there is a trade-off, since there is something to gain in terms of potentialities of the software - it allows to perform more complex tasks - but it is also narrowed the number of possible adopters.

As we can see in figure 3, most of the reviewed systems, whether they use software or not, do not present any information about the type of license in place (if any). This may be a consequence of the fact that they are not widely distributed (?).

From the systems that specify licenses, there is the same amount of proprietary and open-source type licenses, which clearly indicates the co-existence of the two segments on the market. Looking at the timeline, there has been an increased adoption of ”free and open-source-type” licences, and the expression of interest in adopting these licenses in systems that still rely on proprietary components. This is a very positive trend with great social impacts, specially regarding the adoption of these systems in developing countries (see section 4.5).

The table bellow presents the 25 reviewed systems.

Name	**Institution**	**Description**	**Technical Specifications**
Fishframe	ICES	Web- based data warehouse application. It is a platform for data compilation and tabulating, that allows: upload of information, data quality control, data integration and raising.	Modularized software architecture; server client architecture; browser independent; well specified format;

continued on next page

continued from previous page

Name	**Institution**	**Description**	**Technical Specifications**
InterCatch	ICES	Web-based system for handling fish stock assessment data, focusing on documenting characteristics of the catches.	Client-server architecture (runs on a browser); it uses ASP and SQL Server;
ICES Spatial Facility	ICES	Free interface for searching, viewing and downloading spatial data.	WMS service (Geoserver) and client (OpenLayers); it was developed in PHP, with an SQL Server database and makes use of the Geonetwork;
ICES EcoSystem Data	ICES	Spatial database; it uses the standards adopted by the user community for the dissemination, visualization and sharing of ICES raw spatial data.	WFS server and client(Geoserver, OpenLayers). MS SQL Server database.
FMS	Schooner Solutions	Software that allows fishermen to track and manage information on daily trap fishing activities; at the same time, it produces reports of catch and effort for the DO.	Stand-alone software that runs on Windows (98 to XP).

continued on next page

continued from previous page

Name	**Institution**	**Description**	**Technical Specifications**
The Canadian Albacore Tuna Catch And Effort Relational Database	Oceanic Fisheries Program (SPC)	System that provides annual catch estimates for the Canadian albacore troll fishery based on sales slips, logbooks, and hail Information.	MS Access
TUFMAN	Oceanic Fisheries Program (SPC)	Database fishery management tool. It provides solutions for: data entry, data management, data quality control, administration, and reporting.	Server-client architecture; server in SQL Server and client in MS access (supports the free versions); the mapping is provided through a loose couple with Mapinfo;
Catch and Effort Query System (CES)	Oceanic Fisheries Program (SPC)	Menu-driven system, which allows member countries to extract summaries of operational logsheet data, aggregate public-domain catch and effort data, and annual catch estimates.	It uses Mapinfo for mapping.

continued on next page

continued from previous page

Name	**Institution**	**Description**	**Technical Specifications**
Fishery Analyst Online version	MappaMondoGIS	Web-based application, to query fishery data, analyze and visualize temporal and spatial patterns of fishery dynamics.	It uses the Google and ArcGIS JavaScript APIs and the Dojo framework.
Fishery Analyst for ArcGIS 9.x and 10	MappaMondoGIS	Software for the quantitative estimation and visualization of catch and effort and their variation in space and time, analysis of fishing vessel utilization, data quality control, and deriving information on the location of important economic and threatened species.	Developed in VBA (arcobjects), inside ESRI ArcGIS. The data is stored in DBF files.

continued on next page

continued from previous page

Name	Institution	Description	Technical Specifications
FiRST	Collaborative effort across eight South and Southeast (World Fish Center) Asian Countries.	Data management system for scientific trawl survey data. It includes data summary and reporting functionality; outputs: catch, CPUE and biomass estimates. It includes visualization tools, an analytical routine to estimate biomass, and data import/export modules.	Server-client architecture. Server in MS access/ SQL Server (Windows); client in a browser.
PESCART	Instituto Nacional de Investigação Pesqueira (IIP)	Database system for storing data from artisanal fisheries and effectuate some calculus and reports (catch, effort, CPUE). It includes an interface for data acquisition, analysis and reporting.	Developed in Microsoft Access 97 (Windows 98); the system is split in two databases: one contains the data and the other one the app (UI+processing); it uses some MS specific ActiveX (Microsoft FlexGrid Control 5.0) and Libraries (.e.g DAO)

continued on next page

continued from previous page

Name	**Institution**	**Description**	**Technical Specifications**
Geocrust	University of Algarve (UALG)	System that uses Satellite data (VMS) to map effort and landings;	Modular stand-alone application with embedded GIS. It was developed using: Visual Basic 6.0., ADODB and MapObjects 2.0 Pro. It uses a MS Access Database (2000).
Geocrust 2.0 (Geopescas)	University of Algarve (UALG)	Improvement of the Geocrust application, as a consequence of extending the methodology to the ground fish trawl flee, a larger period, and a larger geographical area.	The db was migrated to MySQL running on a Linux server (higher security). It were introduced Artificial Intelligence methods (automatic classification methods) for trawl detection, in addition to the manual procedures.

continued on next page

continued from previous page

Name	Institution	Description	Technical Specifications
Geopescas Website	University of Algarve (UALG)	Website to disseminate geo-referenced information on fishing effort, landings, catches and catch rates, through fisheries researchers, fishery administration and fishing industry. It shows some of the maps generated by the Geocrust software and it offers dynamic mapping, using Geotools.	It uses the Geoserver mapping component, costumised through JAVA programming.

continued on next page

continued from previous page

Name	Institution	Description	Technical Specifications
COST	Project financed by the European Commission under the call FISH/2006/15 – lot 2	Open Source Tool for raising and estimating properties of statistical estimates derived from the Data Collection Regulation.	set of R libraries, with the following functionality: import and handle fisheries data, explore the data , estimate the parameters and related precision and do simulation. It relies on data in a well specified forma.t
Catch Assessment Survey and Statistical frame survey of the artisanal fisheries of Kontagora Reservoir	?	Catch Assessment Survey and Statistical frame survey, that aims to sustain the formulating of management plan and policies for the Kontagora reservoir.	Count and weighting of the fish.
Artisanal Fisheries Data Collection And Management In Tanzania	UNU Fisheries Training Program Fisheries Division - Statistical Unit	Simple sampling procedure, community based data collection model and specifying of types of data to be collected. This system also includes analysis methods and a ”tax motivation” strategy.	MS excel, MS Access Database.

continued on next page

continued from previous page

Name	Institution	Description	Technical Specifications
IFMP	Lake Victoria Fisheries Organization (LVFO)	Catch Assessment Surveys, two-stage stratified sampling design and estimation of CAS-based Indicators.	data forms (paper), MS Excel.
Barahona Data Collection System	Barahona Data Col PROPESCAR-SUR project	National Data Collection System	it relies on paper records and the calculations are done" by hand".
Samaná Fisheries Development Project (CEDEP)	Japan?	Logbook recording system, for a small number of vessels.	?
FISBOAT	IFREMER	Fisheries Independent Survey-Based Operational Assessment Tools. The tools tackle the major issue of unreliable fish stock assessment, through the use of new methods based exclusively on research vessel survey data.	R, FLR simulation framework (R based)

continued on next page

continued from previous page

Name	**Institution**	**Description**	**Technical Specifications**
Probability-based survey techniques in Mozambique	Instituto Nacional de Investigação Pesqueira (IIP)	Probability-based survey techniques for monitoring catch and effort in the coastal small-scale fisheries.	onsite survey, database (?)
Programme for refining estimates of local fishery production in Western Samoa	Peace Corps Marine Biologist Fisheries Division, Western Samoa	A programme for refining estimates of local fishery production, that consists in data collection(interviews) and data analysis;	It uses written down interviews to collect data and a hand-held programmable calculator for analysis; as a database, it is used DBASE III.
Electronic Fish Catch Logbook Project (EFCL)	Northwest Fisheries Centre (NWFSC)	Ship-based software and shore-based web application to collect and report logbook, catch "(fish tickets"), biological sample and observer data. There is also a report system, a central database and web based app to upload data.	Client server architecture, Oracle database, Arcinfo GIS;

continued on next page

continued from previous page

Name	Institution	Description	Technical Specifications

Table 1: Table-resume of the reviewed Fisheries Management Systems

In the following table, the reviewed systems are organized according to their technical characteristics:

- *internet*: indicates if the software needs an internet connection to work (binary value);
- *GIS*: indicates if the software uses GIS (binary value);
- *database*: indicate if the software uses a database (binary value);
- *license*: indicates the type of license;
- *portability*: comments about the cross-platform ability;
- *expertise*: indicates if there is the need of expert knowledge to run the system (binary value);
- *released*: indicates if the system is released and already in use;

Name	internet	GIS	license	database	portability	expertise	released
Fishframe	1	1	BSD	1	Since the clients are browser based, they are cross platform; the server has to run on Windows, since it uses Microsoft specific active technologies (ASP)	0	yes

continued on next page

continued from previous page

Name	internet	GIS	license	database	portability	expertise	released
InterCatch	1	?	?	1	Since the clients are browser based, they are cross platform; the server has to run on Windows, since it uses Microsoft specific active technologies (ASP)	?	?
ICES Spatial Facility	1	1	?	1	Server and client are cross-platform.	0	yes
ICES EcoSystem Data	1	1	?	1	Server and client are cross-platform. Database is Windows specific.	0	yes
FMS	0	?	proprietary	?	Windows only (98 to XP)	?	under final modifications
The Canadian Albacore Tuna Catch And Effort Relational Database	0	1	?	1	Windows only	0	yes

continued on next page

continued from previous page

Name	**internet**	**GIS**	**license**	**database**	**portability**	**expertise**	**released**
TUFMAN	0	1	?	1	Windows only	?	yes
Catch and Effort Query System (CES)	0	1	?	1	Windows only	0	yes
Fishery Analyst Online version	1	1	proprietary	?	The client is free and cross platform (runs on a browser)	0	yes
Fishery Analyst for ArcGIS 9.x and 10	0	1	proprietary	1	Windows only	1	yes
FiRST	1	1	?	1	Since the client runs on a browser, it is platform independent. Server runs on Windows.	0	yes
PESCART	0	0	?	1	Windows only	0	yes
Geocrust	0	1	?	1	Windows only	0	yes
Geocrust 2.0 (Geopescas)	0	1	?	1	The application runs on Windows, but the database can now run on Windows or Linux.	0	yes
Geopescas Website	1	1	?	1	?	0	demo version only

continued on next page

continued from previous page

Name	internet	GIS	license	database	portability	expertise	released
COST	0	1	Open-Source	1	since it relies in R, it is cross-platform.	1	yes
Catch Assessment Survey and Statistical frame survey of the artisanal fisheries of Kontagora Reservoir	0	0	?	?	?	0	yes
Artisanal Fisheries Data Collection And Management In Tanzania	0	0	?	1	Windows only	0	yes

continued on next page

continued from previous page

Name	internet	GIS	license	database	portability	expertise	released
IFMP	0	0	?	0	Windows only	0	yes
Barahona Data Collection System	0	0	?	0	n/a	0	yes
Samaná Fisheries Development Project (CEDEP)	0	0	?	0	?	?	yes
FISBOAT	0	0	Open-source	0	multi-platform	1	yes
Probability-based survey techniques in Mozambique	0	0	?	?	?	?	yes
Programme for refining estimates of local fishery production in Western Samoa	0	0	?	1	Windows only (DOS)	0	yes

continued on next page

continued from previous page

Name	**internet**	**GIS**	**license**	**database**	**portability**	**expertise**	**released**
Electronic Fish Catch Logbook Project (EFCL)	1	1	?	1	Some components run on a browser, so they may be cross-platform, but the database and GIS are Windows only.	0	

Table 2: Reviewed Fisheries Management Systems, according to their characteristics

The next table, goes a bit more in depth, and it analyses some of the advantages and disadvantages of each system.

Name	**Advantages**	**Disadvantages**
Fishframe	- data ownership of the recipients; - deals with incomplete data; - cross platform); - enforces confidentiality; - involvement and engagement of stakeholders; - bottom-up approach	- some components require Windows XP; - development is Windows based; - for complex tasks the browser solution is not efficient;
InterCatch	?	It uses proprietary and platform-specific software
ICES Spatial Facility	- cross platform (browser based); - adopts OGC standards; - friendly UI; - uses Free and Open Source Software(FOSS);	There is no component of data acquiring in this system; it is assumed the raw data is validated.
ICES EcoSystem Data	- cross platform (browser based); - adopts OGC standards; - friendly UI; - uses Free and Open Source Software(FOSS);	- some requests may be slow due to the quantity of data; - - it deals only with survey data;

continued on next page

continued from previous page

Name	Advantages	Disadvantages
FMS	It does not require a very powerful hardware to run.	- it runs only on windows; -it has a cost;
The Canadian Albacore Tuna Catch And Effort Relational Database	- it deals with unreported catch; - although it is a post-season analysis tool, in can be used as in-season analysis (based only on trip logs);	- the results may be biased due to the use of logbooks without saleslip data; - other sources of bias are the ”public sales” and ”take homes” that are not registered anywhere (they ”escape” from logbook and sales slips);
TUFMAN	- the website provides excellent documentation on how-to use the application; - flexible system, able to cater the needs of each country; - it allows to identify gaps in data;	- it uses proprietary components (Mapinfo, Microsoft Office Pro); - it does not feature data acquisition, and therefore it is dependent on the quality of input data; - it does not provide tools to fill the gaps in the data; - it is not found anywhere to download;
Catch and Effort Query System (CES)	- developed reporting facilities, with the use of graphs, tables and maps;	- it uses proprietary components (Mapinfo); - it is not found anywhere to download;
Fishery Analyst Online version	- lightweight and cross-platform application;	-it has a cost;
Fishery Analyst for ArcGIS 9.x and 10	- it takes advantage of GIS to analyze geodata; - it provides a time series animation; - it provides data quality control routines; - it uses; transversal information (economical); - confidentiality is addressed;	- it has a cost, although a demo version is provided for free; - it requires ArcGIS 8.3. or 9 with Spatial Analyst (and it needs this specific versions to work); - iit requires some knowledge of ArcGIS to use; - it requires some detailed georeferenced data to start with (no estimation provided!)

continued on next page

continued from previous page

Name	Advantages	Disadvantages
FiRST	- client is cross platform; - no installation required for the client; - a regional database is “compiled” from the national databases; - outputs can be used for ecological modelling (very useful for management – devise or revise zonation schemes);	- the database is designed for scientific trawl surveys; - it requires installation of a webserver; - the exporter uses mostly proprietary formats (MS Office, Crystal reports, MS Access);
PESCART	- intuitive UI following the information flow; - it is a very complete system, since it covers: data acquisition, estimation and reporting.	- it targets artisanal fisheries only (the basic unit is fishing unit); - ”bulky” setup with a lot of requirements; - the application does not have any access restrictions (no permission system implemented); - the limitations of MS Access ”clash” with the design of the app (e.g. number of fields on a table does not allow the selection of a large number of species;)
Geocrust	- it combines positioning and landing data, from different sources (IGP, DGPA, logbooks); - extense use of GIS, not only for display but also for calculations (e.g.: buffers to define start and end of fishing trips); - it is generic enough, to be used with other fleets/geographic areas/species;	- it is based on proprietary technologies; - it relies on accurate positioning data (VMS); - it was not addressed a permission model for the UI (only superuser);

continued on next page

continued from previous page

Name	Advantages	Disadvantages
Geocrust 2.0 (Geopescas)	- the procedure of identification of trips was enhanced with AI; - DBMS was migrated to a free, multi-platform software;	- it is still based on proprietary technologies; - the AI procedure needs adequate "training", in order to produce good results;
Geopescas Website	- it provides an interface for querying maps; - it exports georeferenced maps, images and XML, that can be integrated in third-party apps.	- it depends on technologies Ggeotools) that are heavy on the browser and require the installation of the Java plugin; - it was only released a demo version;
COST	- space dimension is included in the analysis; - uses well known technology; - promotes import and export of data; - open source and cross platform tool; - interfaces with Fishframe format;	- relies in good quality and well formatted data; - it requires expert knowledge;
Catch Assessment Survey and Statistical frame survey of the artisanal fisheries of Kontagora Reservoir	- it is a complete system, including data collection (survey) and statistical analysis of the results; - it comprehends both effort and catch;	Although the data is collected spatially, there is no indication that this component is taken in account in the sampling and analysis.
Artisanal Fisheries Data Collection And Management In Tanzania	- relatively inexpensive; - it does not need specialized staff to run and maintain it; - it contemplates the community participation;	There is no reference to GIS in the sampling procedure (although the sampling is on space), or in the estimation process.
IFMP	- it provides estimations of the catch over the years, which allows to detect trends, to detect need for further data and to evaluate management interventions.	- since data collection is through papper sheets, there is no data quality control on introduction; - there is no mention of a solution to store and manage the data over time (DB); - there is no use of GIS, although the PSU are spatial;

continued on next page

continued from previous page

Name	Advantages	Disadvantages
Barahona Data Collection System	- the economic variables are well catered; - it gathers relatively accurate catch and effort information based on trip interviews; The effort variables, in particular, seem well thought out and adequate for monitoring CPUE and estimating fishing mortality; - the system was sustained, even after withdraw of external funding;	- mortality and growth cannot be estimated with current data ; - there is the need for a new approach fro estimating discards; - the current sampling procedure does not follow a statistically rigorous method. - there is no database to store and manage data on a long term; - there is the need of training; - stock assessment is not contemplated in this system;
Samaná Fisheries Development Project (CEDEP)	- data collection is rigorous;	- the logbooks are not formally structured; - the data is not maintained in a database;
FISBOAT	- these tools provide robust and reliable scientific advice to fishery managers and policy-makers, including a management model driven by numerically defined harvesting rules; - it is provided an early-warning mechanism, giving policy-makers more time to respond proactively to evolving challenges.	- the methods are based exclusively on research vessel survey data; - it requires expert knowledge to be used (statistics and computation);

continued on next page

continued from previous page

Name	**Advantages**	**Disadvantages**
Probability-based survey techniques in Mozambique	- the probability sampling produces unbiased estimates of catch, effort, and other key parameters; - the precision in such estimates can be quantified; - it is cost-effective; - the underlying theory is well developed and documented;	- no mentioning of use of GIS;
Programme for refining estimates of local fishery production in Western Samoa	- cost-effective method of gathering basic information that provides clues to changes in the fishery as a whole; - it requires a small input of resources for the information obtained;	- the data quality of the survey is threatened by the hiring of causal workers, without adequate supervision; the same for data entry in the database; - outdated database technology; - there is the need to collect more data (from the villages) to estimate the catch; - there is the need to design more complete entry datasheets;
Electronic Fish Catch Logbook Project (EFCL)	- innovative effort to automate and standardize logbook data. - it implements permissions through roles; - it supports secure connections (data encryption) - the use of a software to input the raw data, allows a first validation layer that does not exist in traditional systems using papper sheets; - costumisable;	- it is based in proprietary technologies; - it relies on internet access for data uploads and visualization;

continued on next page

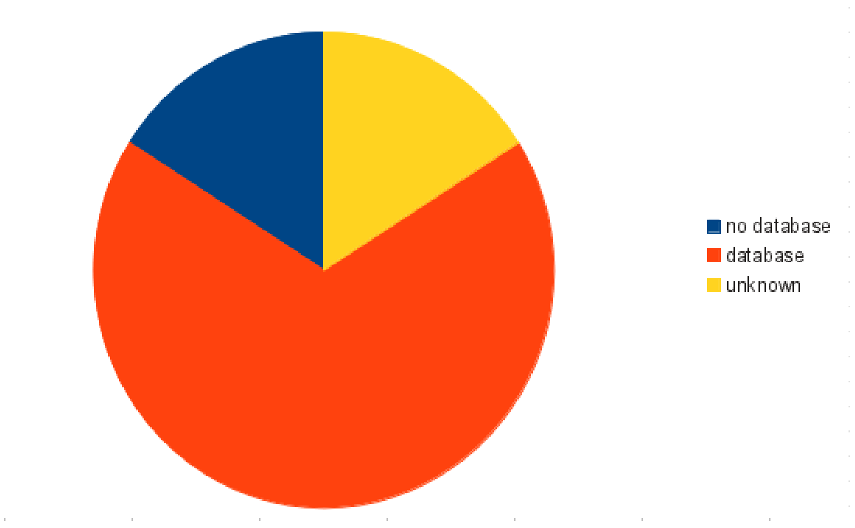
continued from previous page

Name	**Advantages**	**Disadvantages**

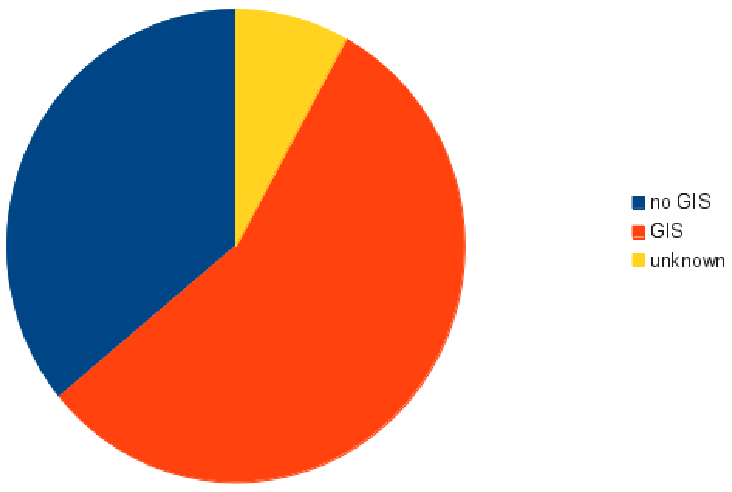
Table 3: Advantages and disadvantages of each system

On section 3 we present a set of fishery management tools that aim to fill some of these "gaps" on existent systems, reusing some of its (good) ideas and adding new technologies that are "giving cards" on other fields (like fuzzy mapping, on section 3.4, or artificial intelligence), keeping in mind the limitations faced by potential adopters (developing countries). Some of these ideas arised during the development of Medfisis-CAS [11], an (almost) full free and open-source system that brings together data from Logbook and Artisanal Fisheries in the same application/database; some of the concepts developed during this project such as abstractability of entities (see section reuse) and configurability of components can be very useful in all aspects of fishery data management systems.

Figure 4 shows some of the challenges that can be addressed by a set of fisheries management tools. We propose a loose couple architecture, where we maximize the use of existing tools: to cut development time, to benefit from their value and also to involve a community of users that can take advantage of their expertise in specialized tools (like for instance R, or QGIS) to push the system further. The modularization of the system also means that we can distribute tasks, and to a certain extent develop them independently without compromising the rest of the system.

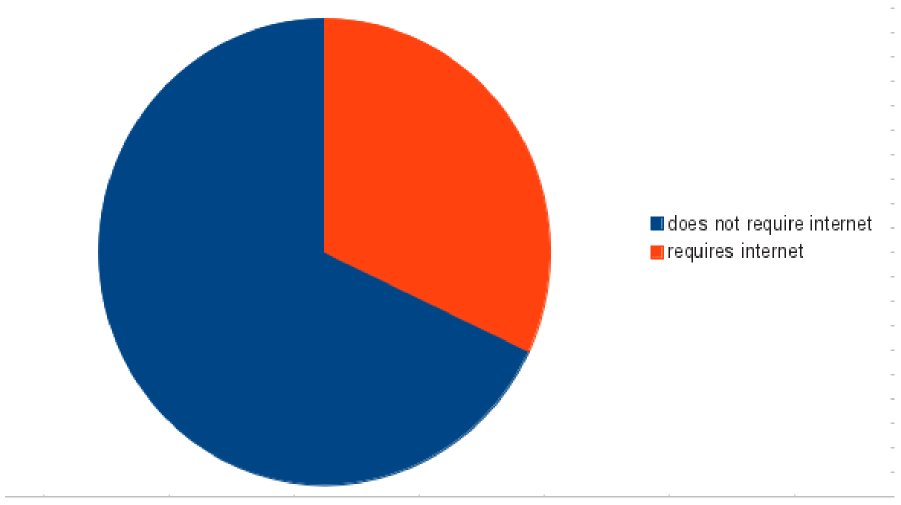


(a) Software use of database;

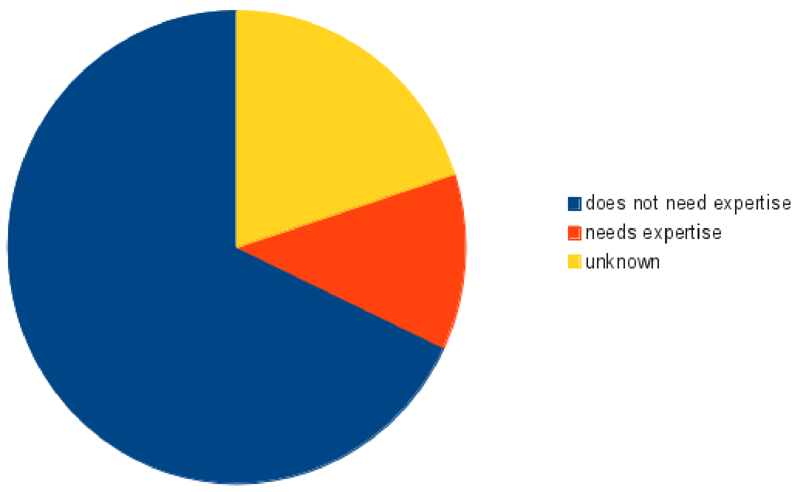


(b) Software use of GIS;

Figure 1: Aggregated analysis of the use of specific software components in the reviewed fisheries management systems;



(a) Need of internet connection, in order to run the software;



(b) Need for expert knowledge, in order to use the software

Figure 2: Aggregated analysis of specific characteristics the reviewed fisheries management systems;

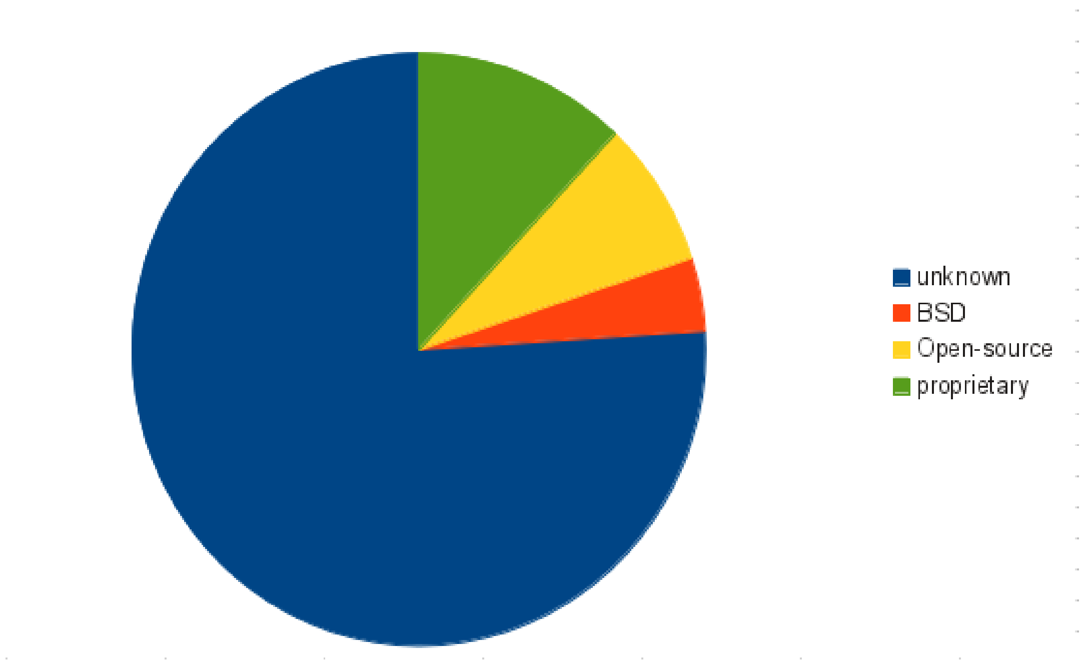
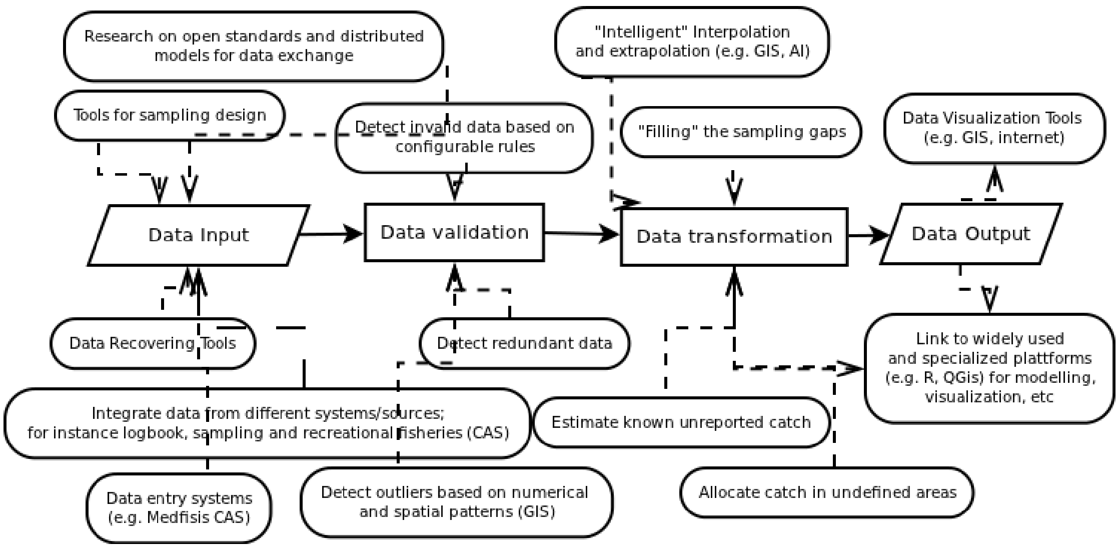


Figure 3: Software licenses;



Research on open standards and distributed models for data exchange
"Intelligent" Interpolation and extrapolation (e.g. GIS, AI)
Tools for sampling design
Detect invalid data based on configurable rules
"Filling" the sampling gaps
Data Visualization Tools (e.g. GIS, internet)
Data Input
Data validation
Data transformation
Data Output
Data Recovering Tools
Detect redundant data
Link to widely used and specialized plattforms (e.g. R, QGis) for modelling, visualization, etc
Integrate data from different systems/sources; for instance logbook, sampling and recreational fisheries (CAS)
Estimate known unreported catch
Data entry systems (e.g. Medfisis CAS)
Detect outliers based on numerical and spatial patterns (GIS)
Allocate catch in undefined areas

Figure 4: Some Interesting Challenges on Fishery Management Systems;

3 Project Objectives

The design of a software system for fishery management, as described in section 2 (figure 4), should contemplate four stages: **input, validation, transformation and output**. The main target for each of these stages is the data about fisheries, in its raw or transformed state (catch, efforts), but it can also be other (complementary) data (meteorological, oceanographical, socio-economic, etc). In the following subsections we identify a serious of tasks that are *on-demand*, and could result in very useful tools for fisheries scientists and managers. In our modular design they can be seen as a first approach rather than a complete description of the system.

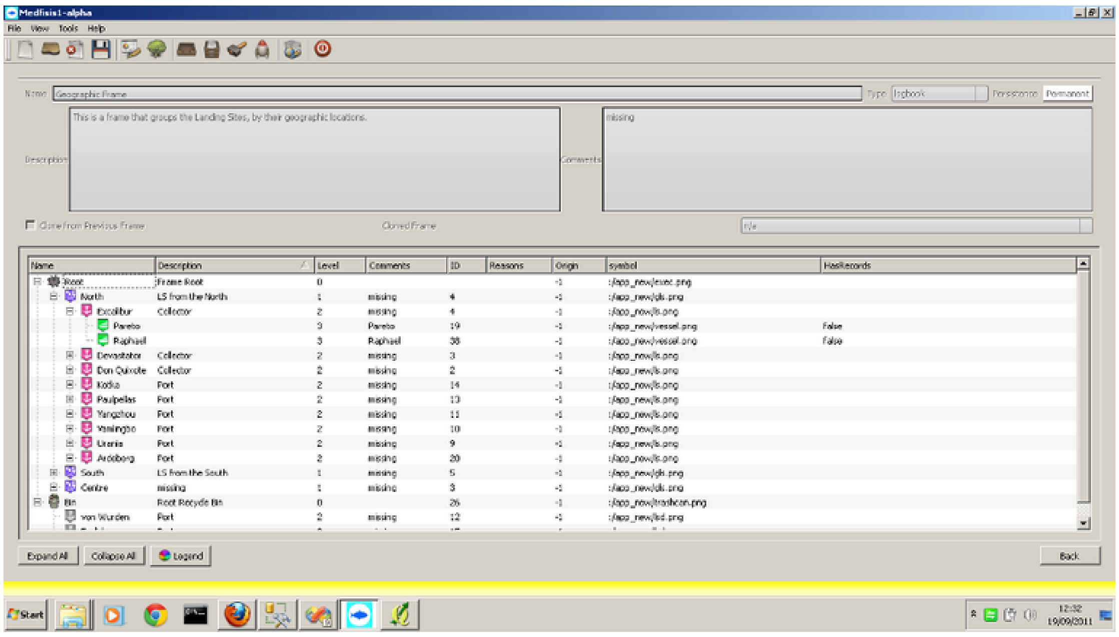
3.1 Data Input

The data input component contemplates the collection of information for the system, either by creating this information "from scratch" or by importing it from different supports/formats.

- **Tools for Sampling Design**: Before even introducing information in the system, it is often necessary to collect this information; the design of an unbiased sampling scheme, that minimizes the costs and maximizes the quality of the results is an important and often overlooked task; overlooked, because often the designer does not possess the tools to make it an easy and efficient process. There are many advantages of using a spatial framework (GIS) on this particular problem, and suggestions can guide the sampler to informed and wise decisions.
- **Research on Open Standards and Distributed Models for Data Exchange:** The problem of exchanging data between different instances of the system is a pertinent and frequent one. JSON [12] is an open and well-known standard that constitutes a "light" alternative to XML. Also, the design and implementation of a server-client model, where we leave the "heavy processing load" to the server and create "light" data acquiring clients (for instance running on mobile phones and tablet PCs) can simplify the task of data integration, as we only update restricted parts of the system. On section 4.2 we discuss the possibility of implementing a distributed system, using a non-relational model.
- **Data Recovering Tools:** There is a large quantity of historical data that is lost in terms of its value, just because it is not in an usable format. Data series take many years to assemble: not only it costs a lot of money to generate new data, but it is not actually possible to recreate past information; thus when this information is lost, it is lost forever. It could be a very rewarding task to at least try to compile this information, that may not be usable for different reasons; hard disks that have bad sectors, obscure data formats whose specification was lost, data in non digital support and nearly illegible: these are all problems eligible for "data archaeology", that can make use of "sophisticated" software techniques such as: writing recognition, deyncription algorithms and hard-disk recover tools.

- **Integrate data from different systems/sources:** The integration of different types of information provides a rich basis for the analysis and transformation of information; this includes having data from different sources such as oceanographical, meteorological, sociological, etc; apart from implementing a system that is able to import different formats (satellite images, excel spreadsheets, vector maps, etc) it is also relevant to design a system that can deal with the nature of this variability - for instance recreational fisheries data, logbooks and sales slips can be integrated with a unified and generic design and provide a richer basis for fisheries statistical analysis (see [11]).

- **Data Entry User Interfaces:** Database clients are a *milestone* in fisheries applications and common examples of this are digital logbooks and fleet register software's (see [11]). They reproduce the behavior of paper sheets for data collection, with all the advantages of data checking, and "feed" directly the database; if well designed and data operators are trained, they are also quicker and more friendly to use than the replaced papper sheets. There is still a room for improvement on this UIs, for instance enabling multiple choices or configurable validation rules [1].

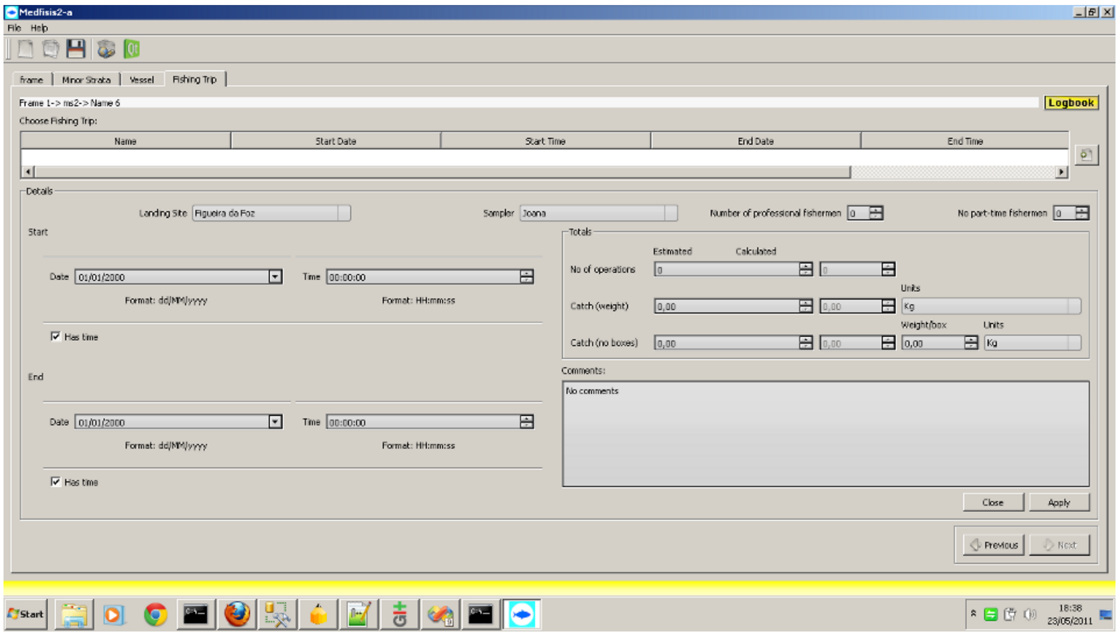


Name: Geographic Frame
Type: logbook
Persistence: Permanent
Description: This is a frame that groups the Landing Sites, by their geographic locations.
Comments: missing
Clone from Previous Frame
Cloned Frame: n/a

Name	Description	Level	Comments	ID	Reasons	Origin	symbol	HasRecords
Root	Frame Root	0				-1	:/app_new/exec.png	
North	LS from the North	1	missing	4		-1	:/app_new/gls.png	
Excalibur	Collector	2	missing	4		-1	:/app_new/ls.png	
Pareto		3	Pareto	19		-1	:/app_new/vessel.png	False
Raphael		3	Raphael	38		-1	:/app_new/vessel.png	False
Devastator	Collector	2	missing	3		-1	:/app_new/ls.png	
Don Quixote	Collector	2	missing	2		-1	:/app_new/ls.png	
Kodka	Port	2	missing	14		-1	:/app_new/ls.png	
Paulpellas	Port	2	missing	13		-1	:/app_new/ls.png	
Yangzhou	Port	2	missing	11		-1	:/app_new/ls.png	
Yaningbo	Port	2	missing	10		-1	:/app_new/ls.png	
Urania	Port	2	missing	9		-1	:/app_new/ls.png	
Ardoberg	Port	2	missing	20		-1	:/app_new/ls.png	
South	LS from the South	1	missing	5		-1	:/app_new/gls.png	
Centre	missing	1	missing	3		-1	:/app_new/gls.png	
Bin	Root Recycle Bin	0		26		-1	:/app_new/trashcan.png	
von Wurden	Port	2	missing	12		-1	:/app_new/lsd.png	

Figure 5: This *tree-like* widget is a tool developed in Medfisis [11] to assist in the design of a sampling frame; the user can drag and drop elements from one group to another, which can be an interesting feature for designing sampling schemes;

[1]As an example, some of this features were already implemented in CAS Medfisis [11]



Medfisis2-a

Frame 1-> ms2-> Name 6

Choose Fishing Trip:

Name	Start Date	Start Time	End Date	End Time

Details

Landing Site: Figueira da Foz
Sampler: Joana
Number of professional fishermen: 0
No part-time fishermen: 0

Start
Date: 01/01/2000 (Format: dd/MM/yyyy)
Time: 00:00:00 (Format: HH:mm:ss)
☑ Has time

End
Date: 01/01/2000 (Format: dd/MM/yyyy)
Time: 00:00:00 (Format: HH:mm:ss)
☑ Has time

Totals

	Estimated	Calculated	Units / Weight/box	Units
No of operations	0	0		
Catch (weight)	0,00	0,00	Kg	
Catch (no boxes)	0,00	0,00	0,00	Kg

Comments:
No comments

Figure 6: Screenshot of a data entry form for fishing tip data, on CAS Medfisis [11]; on this system, forms were designed to accommodate at the same time logbook and artisanal fisheries data;

3.2 Data Validation

Since the data entering the system may arrive from sources other than its data generator interfaces[2], it is important to control the quality and validity of information we are inputing; this step is fundamental for all the following ones, since "bad quality" data will compromise the entire system ("garbage in, garbage out").

The Fishframe application [1], developed by the Danish Institute for Fisheries Research, presents an interesting and complete multi-step validation layer; on a first stage it is performed a basic check, looking at the structure of the data, ranges of values and duplicate values; on a second stage, a more complex check searches for the correctness of the dependency between fields; finally, the last check (which is not compulsory) performs a global analysis, looking at the general patterns in order to identify outliers. Apart from this, it also allows some customizable checks to be introduced by the user, that can be performed automatically. This flexibility and completeness of the validity step, inspired us to identify the following tasks in the data validation component of this system:

- **Detection of redundant data**: The detection (and consequent removal) of duplicate records is an important task, in order to have a "clean" database. It is not

[2]The data entry interfaces should have already their own validation routines removing, at least partially, the need for this step.

complicate to identify automatically records that are **exactly** identical, but the problems arise when we have for instance spelling mistakes, and records that are *almost* the same, but not exactly the same. As this situations are dubious, and may correspond or not to duplicate data, they require user supervision to confirm it; however, there is no reason why they can not be identified and presented to the operator has "possible duplicates"; for this task, it is very useful to use data mining techniques such as a fuzzy approach to language [14].

- **Detection of invalid data based on configurable rules**: Generic rules can be applied to check for the correctness of the data types, correctness of keys, etc, directly based on the database schema. However, some rules such as the format of strings, ranges of values, may change from system to system (for instance according to the geographic location) and the option to create them should be left to the user. To avoid "hard-coding" of rules, which would require a programmer's intervention and consequently difficult the process of exploratory analysis, we propose an interface for user configurable rules. This interface can be similar to a query builder such as the one represented in 7, which may be familiar to some users.

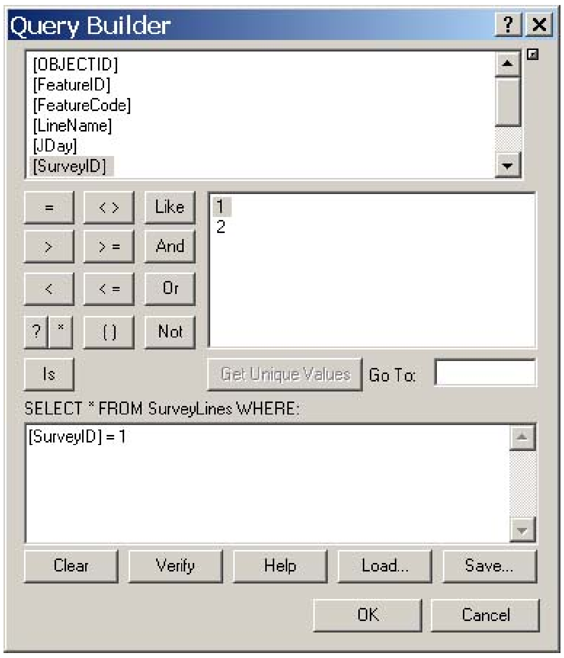
- **Detection of Outliers**: Outliers may arise due to "careless" data acquiring, instrument malfunction, wrong data processing routines, or any other reasons [15]; generic methods based on tabular data (for instance using a principal component analysis) can be used to perform this task, but there is also room for other techniques such as data mining, or GIS; for data with a spatial distribution (which is often the case in fisheries) GIS can provide a very valuable tool for capturing the general spatial pattern and consequently the values that fall outside this pattern (outliers)(see figure 8). After detecting and removing outliers values, we are left with the problem of imputation of "missing values" to replace the removed information (see section 3.3)

3.3 Data Transformation

Data transformation consists in using the information in the system and apply transformations to it, in order to generate new information. These transformations may be used to estimate values extending the scope of the dataset, or to generate indicators that characterize the dataset, like for instance the CPUE, or both. In this system we suggest facilitating the export of data and leave (some) more complex data analysis to specialized software such as R (figure 9) or QGIS (figure 10).

Bellow are examples of some specific (and common) problems with catch data, for which we propose to develop some solutions in the system.

- **Imputation of missing data**: Under-sampled or non sampled strata lead to an unknown bias, a problem that should be reduced with a good sampling design [16]. For the situations where we have to deal with missing values, it is possible to inmputate them, having in mind that also the method used for this operation can



SELECT * FROM SurveyLines WHERE:

[SurveyID] = 1

Figure 7: Query builder from ArcGIS; source: `http://woodshole.er.usgs.gov/pubs/of2006-1381/html/fig19.html`;

introduce some bias in the system. According to ICES [16] automatic methods should be avoid, in favor of expert knowledge, but they can be aided by spatial modelling techniques having in mind that to be use with success the spatial distribution should remain stable over time; this is a cue for developing GIS based models, that can provide the expert with a good basis for informed choices.

- **Allocate Catch in undefined areas**: On the Canadian Albacore Tuna Catch and Effort Relational Database [6] it is presented the problem of having sales or logbook catch data, whose geographic area is undefined. Instead of discarding these values from the calculations of catch by area, the authors found ways to distribute the catch and effort, following the overall geographic pattern of fleet; they allocate these values in proportion with the total catch and vessel distribution and for this purpose GIS can be very useful, for instance to produce a grid, or a contour map (see figure 11) with frequencies.

- **Estimate known unreported catch**: Another delicate problem arises from the incomplete vessel count. If there is an unreported effort than it will probably be reflected in an unreported catch. For the cases where it is known that an unreported effort exists, the authors of the the Canadian Albacore Tuna Catch and Effort Relational Database [6] suggest to extrapolate the total catch estimates based, on estimated number of unreported vessels. According to [17], unreported

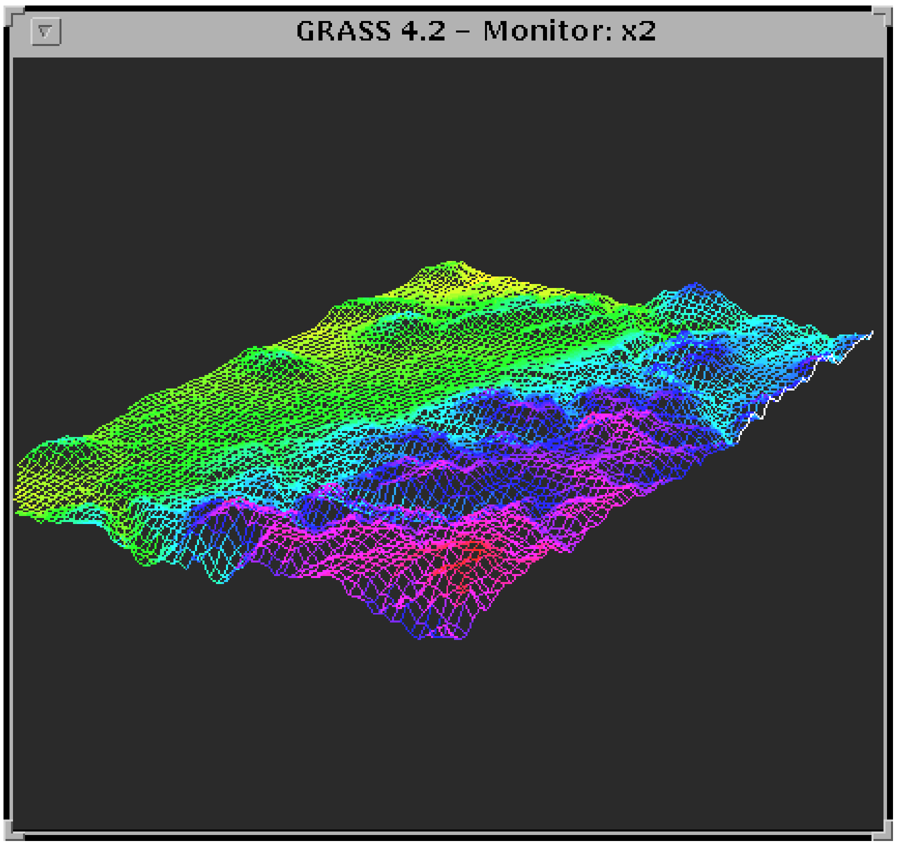


Figure 8: A Digital Elevation Model (DEM) like the one above, can be used to generate a surface of values in order to detect outliers; source: `http://www.grass-kr.org/research/demo1/index.html`;

catch can be as significant as 50% of the total catch and therefore it is important to find suitable methods to approach this problem, like for instance the Monte Carlo Simulation (see figure 12).

3.4 Data Output

Data output, of raw or transformed information, is a component where it is possible (and desirable) to facilitate the link with other platforms by using "friendly" and well-known formats. This could include formats for tabular data (for excel, R, etc), but also formats for spatial data (shapefiles, GeoJSON,etc).

GIS provide some very-rich and expressive visualization tools with a great potential for fisheries data, that we will discuss summarily in the next few paragraphs.

Web mapping is the process of designing, implementing, generating and delivering

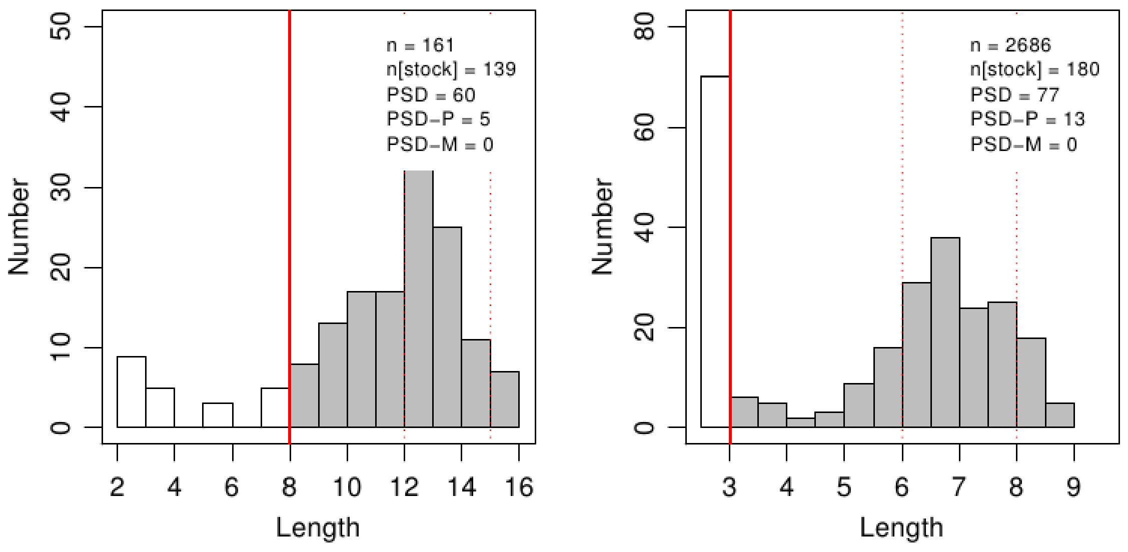


Figure 9: This analysis of the size structure was created using the R package; source: `http://www.ncfaculty.net/dogle/fishR/index.html`;

maps on the World Wide Web [3]. Compared to "traditional" maps, maps on the Internet have many other advantages such as "live updating", and link with many other sources of information; they provide the same interactivity as a GIS software, with the advantage of not requiring a specific software client, since they can run on an Internet browser. Web maps are normally incorporated in portals together with other types of information, the case of ICES Ecosystem Data [18] (see figure 13), but they can also be embedded in a software application (see figure 14).

Continuous maps showing densities, can be created using grids or contours or other methods such as TINs. Grids (such as the one on figure 13) are very common to represent distributions of fishing effort or catches; On the Geocrust project (see [19]), the generation of grids with different sizes allowed the user to have a more generalized or more detailed perception of the fishing effort, which calls its attention to different properties (see figure 15).

GIS functions for cartography are generally based on boolean logic: that is, sharp frontiers between entities. However, when there is uncertainty and vagueness attached to a certain phenomena, this "false" precision may be awkward or simply inadequate (see [20]). The Zadeh's fuzzy set theory is an alternative to this boolean logic and has been proposed has a foundation to GIS design, that can be incorporated to cartographic display of phenomena with a certain imprecision associated to them. On figure 16 we see the comparison of a traditional map, with sharp boundaries and a fuzzy map.

On fishery surveys, where often spatial data relies on human description (for instance a fishermen describing where he was fishing), this could be a very promising approach.

[3]`http://en.wikipedia.org/wiki/Web_mapping`

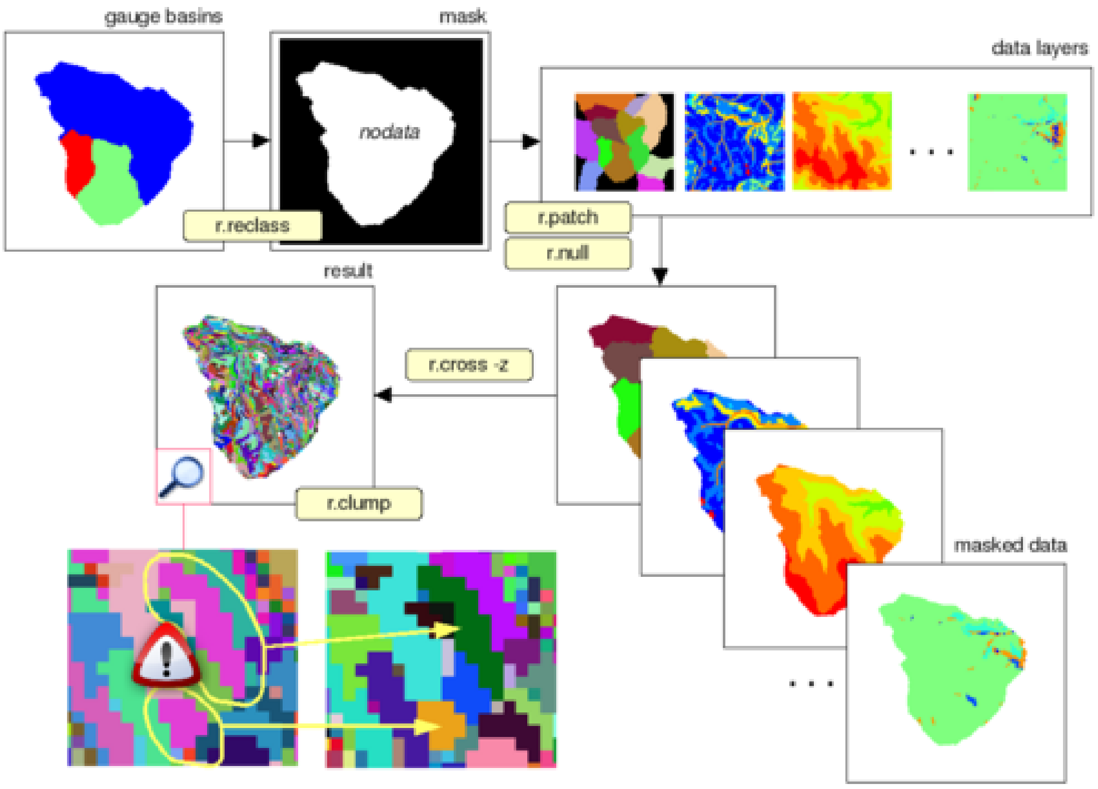


Figure 10: QGIS (in this example using the grass functionality) can be used to overlay different types of information; source: `http://geoinformatics.fsv.cvut.cz/gwiki/Deriving_Hydrological_Response_Units_\%28HRUs\%29_using_a_Web_Processing_Service_implementation_based_on_GRASS_GIS`;

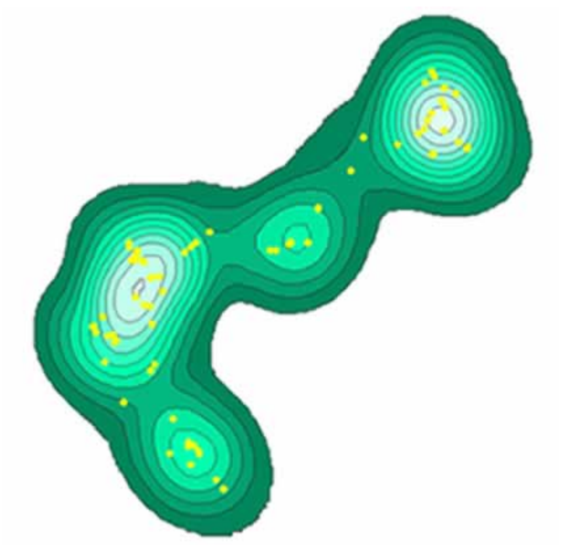


Figure 11: The Kernel Home Range Method is a probability measurement that outputs contours, based on punctual observations; source: `http://www.bio.davidson.edu/people/midorcas/GISclass/GISprojects/grayson/GISMethods.htm`;

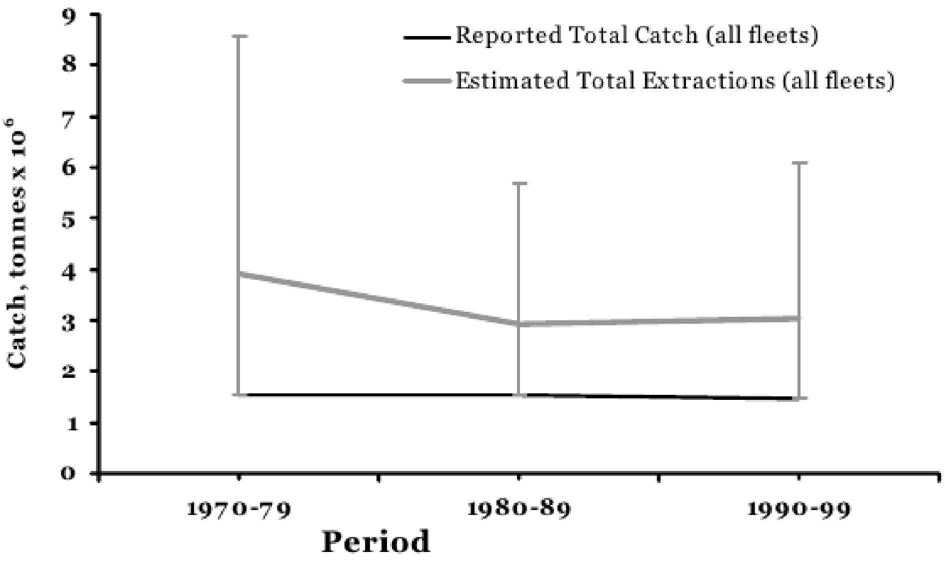
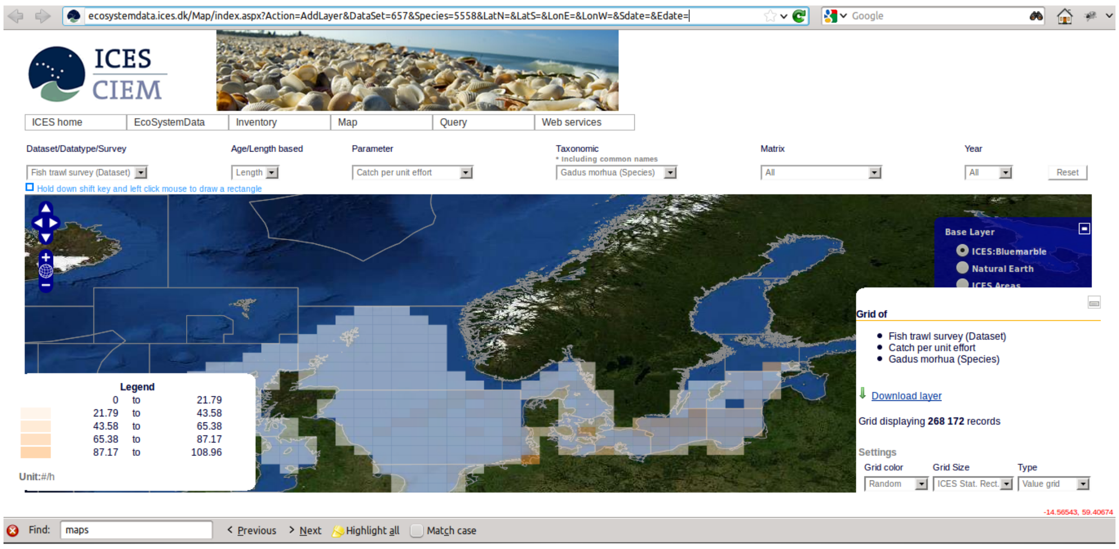


Figure 12: In this example [17], Monte Carlo Simulation was used to elaborate estimates of total extractions of all species from Moroccan waters; in the image we see the comparison with the total reported catch;



Dataset/Datatype/Survey: Fish trawl survey (Dataset)
Age/Length based: Length
Parameter: Catch per unit effort
Taxonomic (* including common names): Gadus morhua (Species)
Matrix: All
Year: All
Reset

Hold down shift key and left click mouse to draw a rectangle

Legend

	0	to	21.79
▨	21.79	to	43.58
▨	43.58	to	65.38
▨	65.38	to	87.17
▨	87.17	to	108.96

Unit:#/h

Grid of

- Fish trawl survey (Dataset)
- Catch per unit effort
- Gadus morhua (Species)

Download layer

Grid displaying **268 172** records

Settings

Grid color	Grid Size	Type
Random	ICES Stat. Rect.	Value grid

Figure 13: Web map of a ICES fish trawl Survey dataset, overlaying a grid of CPUE on top of a layer of Marble (Nasa Worldwind) [18];

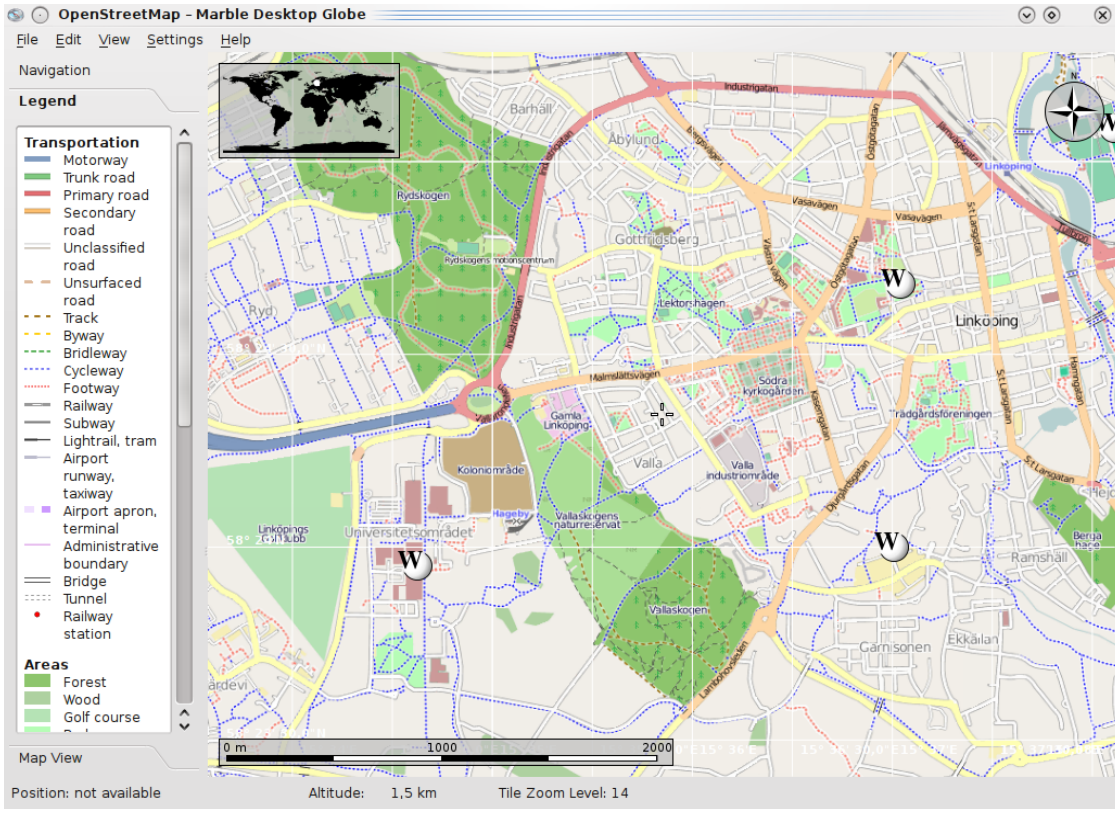
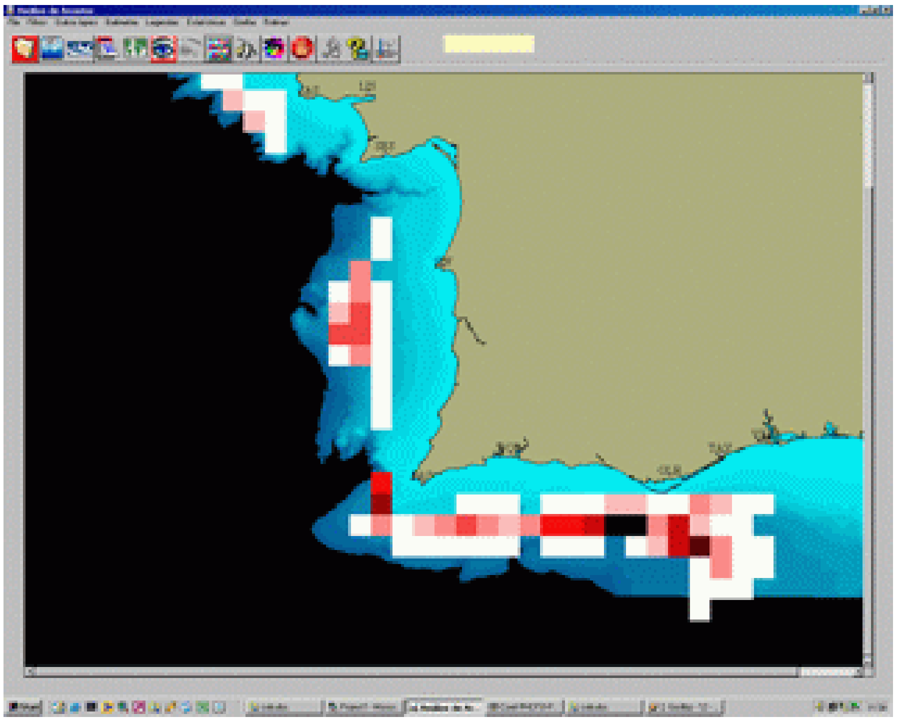
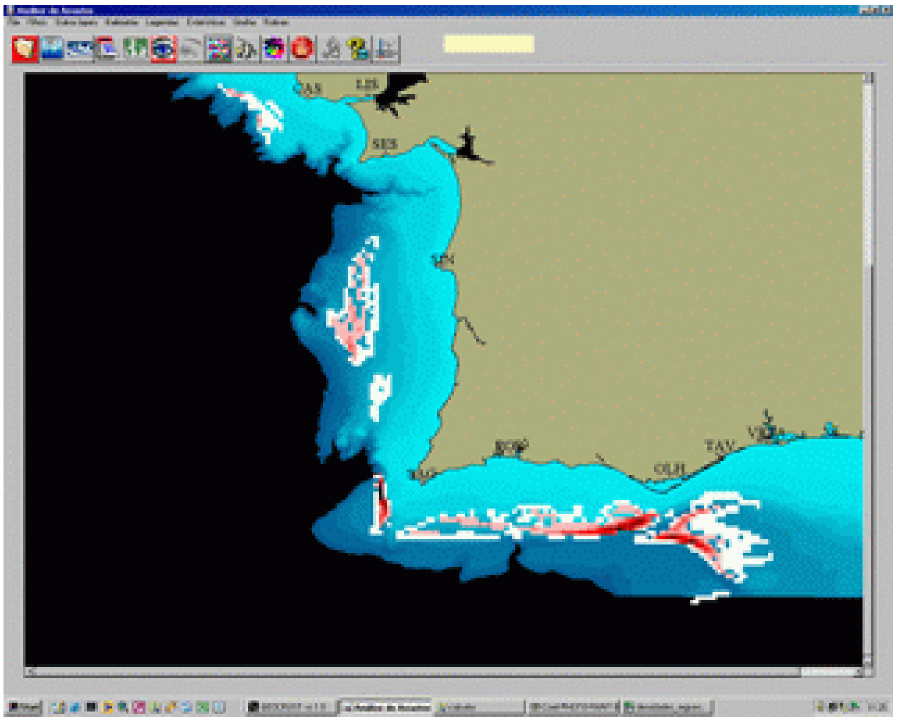


Figure 14: Marble [4] is a free and open source application that displays webmaps, along with other types of maps in a C++ application; source: `http://edu.kde.org/marble/screenshots/generic/marble-linkoeping.png`;



(a) Density map of fishing effort, using a large grid;



(b) Density map of fishing effort, using a small grid;

Figure 15: Fishing effort maps generated by GeoCrust 2.0. To produce these maps the user can choose one or more vessels for a given period of time (quarter, semester, and year); VMS data from valid fishing trips trawl hauls. Density grid size set by the user (minimum 0.06 x 0.06 nm);

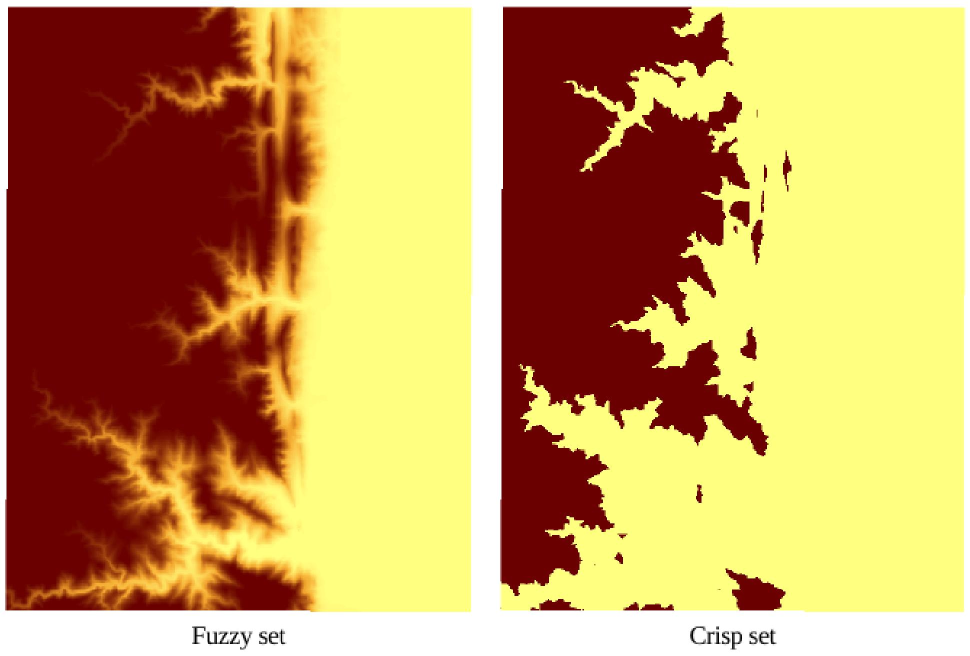


Figure 16: This is an illustrative example of an analysis with a fuzzy logic approach (on the left) and a crisp approach (on the right) [21];

4 Key Aspects of the System

In this section we describe key aspects of the system, that implements the tools listed on section 3. These recommendations are a result of the review on 2.1, and also of the "hands-on" experience of developing fishery data management systems. Some of these aspects may be innovative in the fishery systems context, although established practices on the software development field. Other aspects may be a reaction towards negative practices that we experienced or, on the contrary, positive trends that we wish to support.

Often behind "failure" of software projects there is a disregarding of these practices, rather than technical incapacity. We understand by success, not only having a **good quality, working software, delivered on time, but also to have a system that will continued to be used, maintained and improved by the community**.

4.1 AGILE Approach

AGILE software development [35] is a group of software development practices based on iterative and incremental development; in this framework, requirements and solutions evolve through collaboration between self-organizing, cross-functional teams. There is an emphasis on software quality, adaptation to change, and involvement of team members, that is resumed in the twelve principles of the AGILE manifesto:

- Customer satisfaction by rapid delivery of useful software.
- Welcome changing requirements, even late in development.
- Working software is delivered frequently (weeks rather than months).
- Working software is the principal measure of progress.
- Sustainable development, able to maintain a constant pace.
- Close, daily co-operation between business people and developers.
- Face-to-face conversation is the best form of communication (co-location).
- Projects are built around motivated individuals, who should be trusted.
- Continuous attention to technical excellence and good design.
- Simplicity- The art of maximizing the amount of work not done - is essential.
- Self-organizing teams.
- Regular adaptation to changing circumstances.

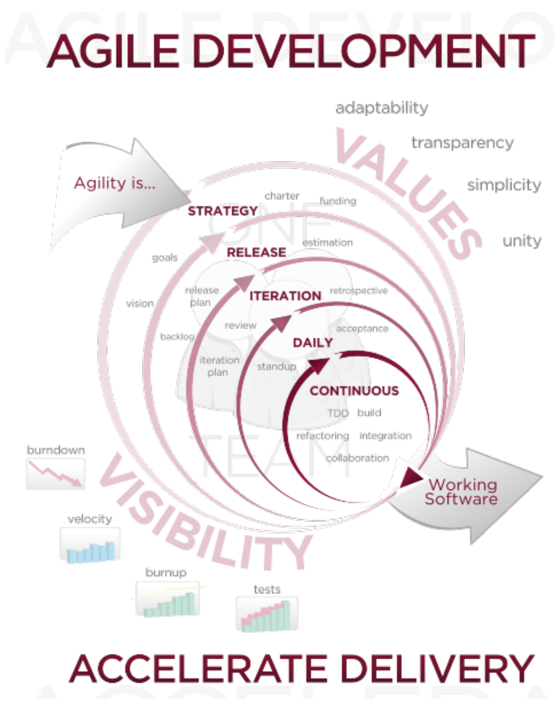


Figure 17: Agile Software Development methodology;
source: `https://en.wikipedia.org/wiki/Agile_software_development`

In a "traditional" approach for software development, e.g. the *Waterfall* model, "change" is a threat to the completion of the project. After "closing" the requirements stage, any changes in the design of the software are counter productive, and will inevitably slow-down the whole process (or condemn it to failure). However good ideas come usually in the late stages of the traditional development cycle, when we already saw some results. For instance, we may create separated forms for introducing logbook and sampling data. However, when we are "demoing" the program we may actually find out that they are quite similar, and probably we could reuse most of the form (and the storage system as well!). We are close to the delivery date, and this change would imply refactoring the entire system, so we have to choose between delaying everything or disregarding the "new idea". With this approach, software systems often lack creativity. AGILE methodology instead makes a point of embrassing change, as something positive.

SCRUM [38] is an AGILE development framework, focused on project management institutions where it is difficult to plan ahead; this is often the case of cross disciplinary projects, involving many different stakeholders.

According to [39], the SCRUM team roles are: product owner (PO), SCRUM master and developing team. The PO is the person who knows everything about the product - he represents the stakeholders; he is also the person responsible for creating and maintaining the *product backlog*. The backlog, can be regarded as a "wish list", or a collection of use cases with everything we want the application to do. The backlog may, and should be adjusted during the development cycle. The PO should have full authority over the management of the backlog. The backlog allows to use interesting progress tools, such as the "burn down chart" where we track the progress of the project, and are able to forecast release dates, based on the "burn down velocity".

A *Sprint* is a short development cycle (weeks to months, depending on the project) where a series of selected items from the backlog are considered "done". The responsible for turning backlog items into incremental development during the the Sprint, belongs solely to the development team and they should have full decision power on "how" they are going to do it. In SCRUM teams there is no project manager, but the equivalent to it would be the SCRUM master. He helps everyone to change the interactions to maximize the value created by the Scrum Team. The SCRUM master interacts with the development team, PO and organization.

The interactions between team members are very important and they are one of the things that makes AGILE projects adaptable. On SCRUM, we have a "daily SCRUM" which is a very short meeting among the development team, to track progress and tackle possible difficulties. At the end of each Sprint there are meetings involving the entire team: a "Sprint review" and a "Sprint retrospective"; the review is focused on the product, while the retrospective is focused on the process. During the review, the SCRUM master marks items as "done", where "done" does not mean software that "looks like it is working", with disorganized code and without testing, but it means "ready to be released", including documentation and testing. After this, the items for the next Spring are defined.

SCRUM **does not solve the development problems**, but it makes them visible and enables targets, with short cycles, that leave room for experiments.

According to [40], the application of SCRUM methodologies has brought noticeable improvement in: productivity, moral of the team, adaptability, responsibility, collaboration and cooperation; in places such as: Yahoo!, Microsoft, Google, Lockheed Martin, Motorola, SAP, Cisco. In the following lines, we list a set of SCRUM recommendations that could be very valuable on the context of this project:

- **Continuous integration and often releases**: This could be achieved by using a versioning system (e.g.: Git) and automated buildings (e.g. every night), with automated generation of documentation (e.g.: Doxygen); using these tools the software would be ready to "release" at any moment (e.g., when decided by the PO), with minimum effort.

- **Cross-functional, self organized development teams**: Till a certain extent there could be some specialization of some members of the development team (for instance in "databases"), but is proved to be much more productive when everybody can do all work, and transfer knowledge; in this way during the Sprint, it is possible to concentrate the effort of the team in the parts where it is more necessary; it is also less "damaging" for the project, if a member of the team is absent, or if he decides to leave.

- **Test team**: There is an exception to the previous point, which is the test team. It is desirable that the members of the development team can also be testers, but this should not replace the role of an independent tester. Professional testers, design tests that the SCRUM team would not imagine or did not have the hardware to implement. They play the role of the "users" and access the system in exactly the same way that users will. The advantage of using a tester rather than stakeholders, is to reduce the testing period by using an efficient methodology.

- **Flexible, "bottom-up", decisions regarding adopting of software frameworks and tools**: The decisions about development should be left to the people who are going to implement it; there should be no hierarchies within the development team (there is no such thing as "lead developer"); everybody is labelled as "developer".

- **Size of the development team**: While there are stages when it is useful to program alone, it has been observed that fewer than three development team members decreases interaction and results in smaller productivity gains [38]. The minimum size of development team has been set to three. Part-time members are something to avoid, as it is important to have their fully commitment during the Spring.

- **Sustainable rhythm of work**: It has been observed that when developers are not driven to work long hours, they are more "energetic" in their work, and often more creative [41].

- **Use of incremental design**: Start with a simple working system and add features improve, and refactor, instead of choosing very complicated designs in the beginning, that will "freeze" the entire system for a long time.

- **Adaptability to change**: in order to face change, there should be flexibility in at least one of these three components: price, time or functionality.

- **Meet often**: In the case of geographically distributed teams, it is possible to promote meetings using the internet. Skype can be used for conference calls, and forums or chats can support discussion; From time to time, it is also useful to bring the team together in one location.

- **Involvement of the stakeholders**: A stakeholder is anyone who is a direct user, indirect user, manager of users, senior manager, operations staff member, the "gold owner" who funds the project, support (help desk) staff member, auditors, developers working on other systems that integrate or interact with the one under development, or maintenance professionals potentially affected by the development and/or deployment of a software project; it is important to engage as many stakeholders as possible in the backlog review; one way of keep them informed (and involved) is the use of a wiki page for discussion, and to present results (as an example, see [36]).

- **Frequent refactoring with emphasis on non-functional characteristics of the software**: For instance: expressiveness of the code, re-usability, etc; the use of good design patterns will ease the debugging and the development of new features, and will increase the collective ownership of the code; software that did not focused on non-functional characteristics can quickly become obsolete, defeating the purpose for which it was built in the first place.

To summarize, the key benefits of embrassing AGILE practices are: **ability to meet deadlines, ability to face "changes", satisfaction across the community** and ultimately, **good quality software**.

Based on the ideas exposed above, we propose an AGILE/SCRUM project team of four people and three roles:

- One product owner;
- One SCRUM master, that also doubles as developer;
- Two multi-functional developers, that also double as testers;

Temporarily we may also need to add a "supporting cast", that helps the PO and the development team to perform specialized tasks (see figure 18).

- **Technical experts**: Temporarily the team may need the help of technical experts to help them overcome a difficult problem, and to transfer their skills to one or more developers on the team (e.g.: database expert, GIS expert).
- **Domain experts**: As the PO represents a wide range of stakeholders (not just end users), it isn't reasonable to expect him to be an expert at every single nuance of the domain. Thus the PO may sometimes bring in domain experts to work with the team (e.g.: African small-scale fisheries expert, small-scale fisheries statistical systems expert).
- **Independent tester**: Effective AGILE teams often have an independent tester working in parallel, that validates their work throughout the software lifecycle.

4.2 Alternatives to the Relational Model

The reviewed systems on section 2.1 have one thing in common: they use a relational database. The relational model was developed by Edgar F. Codd in 1969 [31], as a declarative method for specifying data and queries. It is very established in the software world, it has a strong theoretical (mathematical) basis behind it, and a large number of experienced users that are able to provide a good support. However "antiquity" and "reputation" do not mean that this model is suitable and should be adopted for every situation, as it is often the case.

The relational model is a top-down approach, where we "know everything" about our data. While this may be true for some cases, it is probably not the case of fishery

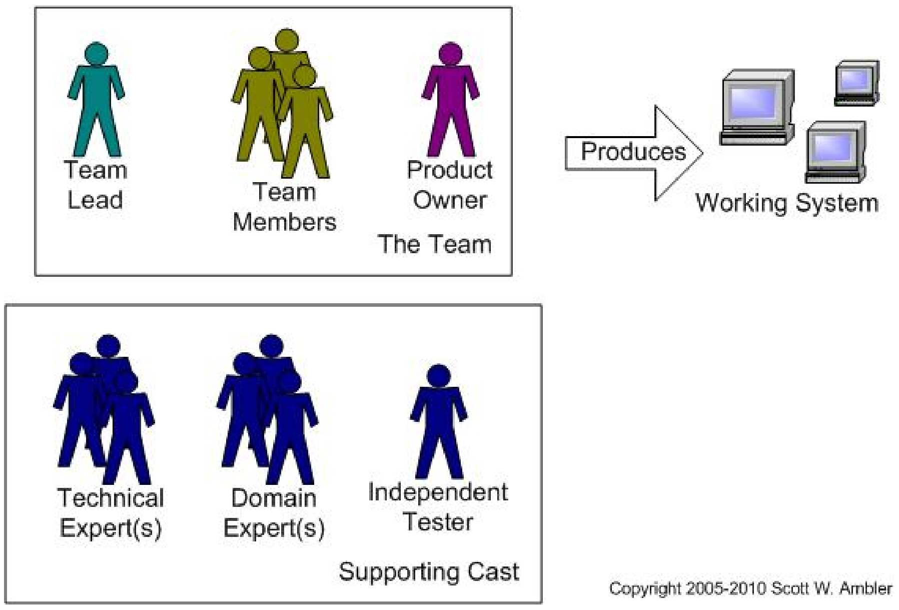


Figure 18: Organization structure of a small agile team; on a SCRUM team, the team lead is replaced by the SCRUM master;
source: `http://www.ambysoft.com/essays/agileRoles.html`

data management systems, where often requirements change over time. For instance a logbook sheet, may include a couple of fields at first, but with user experience we may decide that we want to readjust it and remove some obsolete fields, and add new ones. We may also decide to readjust it because we want to integrate a logbook from another country, that has a completely different structure. Although it is possible to "refactor" a relational database , it is always a "painful" and time consuming experience, specially if it already contains data that needs to be migrated. This is because relational databases rely on a "strong" view about "what" the data should be (the schema), that is more important than the data itself . They are not flexible and they are not AGILE (see 4.1). Once we accept that we do not "know everything" about our data,

than perhaps we should look for a model that complies to this fact.

NoSQL [32] is a broad class of Database Management Systems (DBMS) that do not adhere to the relational model. In the NoSQL model the ability of storing and retrieving large quantities of data is more important than the relationships between elements. These are some interesting features of NoSQL databases:

- They are a good fit for object-oriented programming models; for instance, they enable to represent hierarchical structures such as trees.
- They are designed as distributed systems, which means they scale very well. Their ability to store, manipulate and analyze ever increasing volumes of data can be

maintained without increasing the computational power (more processors or hard-drives); we just need to add more nodes to the system, instead.

- They support data replication, to ensure high-availability and support disaster recovery. This means the the "merging" of databases is directly supported by the system.

The CAP [34] theorem, which applies to many sorts of distributed systems - such as NoSQL databases - states that it is impossible for a distributed computer system to simultaneously provide all three of the following guarantees:

- Consistency: All nodes see the same data at the same time.
- Availability: 100% of requests are completed successfully.
- Partition Tolerance: Any given request can be completed even if a subset of nodes in the system are unavailable.

Many NoSQL systems, "sacrifice" consistency, in exchange of availability and partition tolerance. These systems are called "eventual consistent" [33], which means that given a sufficiently long period of time over which no changes are sent, all updates can be expected to propagate eventually through the system and all the replicas will be consistent. NoSQL systems do not need to be "eventually consistent", but they may choose to do so.

Finally it is important to consider the cases where we wish to achieve scalability, without "sacrificing" consistency and transactions. For those situations it is perhaps pertinent to consider an hybrid solution, using a relational database to store the relations and a NoSQL to store the actual data. There are also some hybrid SQL-NoSQL solutions on the market, that combine advantages of both systems; these are called "NewSQL" databases.

4.3 Redefining the Centre of the Application

One common mistake when designing an application is to put the database or the framework as the centre of the application; e.g.: when designing fishery management systems, it is very common to start by choosing the technology (e.g.: .NET Framework, Oracle) or to design the database before deciding anything else (including, if we have the "need" for a database). This is wrong, as the centre of the application - the highest level and most visible architectural entities - should always be the "use cases" [29]. The first thing should be to know "what" exactly are the data entities involved (the "actors"), how they are related, and how they are used. Only after clarifying these points it is possible to make decisions about a data model, and fit it into a database. Getting a database "involved earlier on" will warp the design, and influence our ideas about what we want to achieve. For instance, the relational model will bias our thinking into a "relational" view of the world, where data is organized into tuples and relations [31].

Databases and frameworks are technical details that may be decided (and changed) later, but a correct design is a crucial - and often disregarded - step in software development.

In an AGILE approach (see section 4.1), developing use cases is regarded as an iterative process; the process should rely on work and refinement, rather than getting them at perfection in the first instance [30]. The primary use cases should be the "sunny day" scenarios; these are scenarios that describe what happens when everything "goes well". These scenarios are very important, firstly because they account for what happens in most of the cases, and secondly because they will help us write the edge cases, or "Rainy day" scenarios.

4.4 Identifying Reuse Opportunity for Use Cases

When designing the system, and particularly when writing the "use cases", it is pertinent to identify opportunities for generalization. Object oriented languages provide a good framework for generalization through "inheritance".

For instance, we may have different types of landing sites, such as "Ports" or "Freezer Boats". If we create an entity "Port", later we will have to create another entity "Boat" copying some of the properties of the "Port" (e.g.: name). If we create instead an object "Abstract Landing Site", we can derive from it entities "Port" and "Boat", ihneritating common characteristics and adding specific ones (for instance the port may also have geographic coordinates, and the boat may have a license plate). Moreover we don't need to refactor any code that referred to generic properties of the "Port", when creating the "Boat" entity, and later we may decide to introduce other entities that we did not planned in the beginning, such as "Virtual Landing Sites".

The benefits of generalization are that in one hand we eliminate duplicate behavior and attributes, but most importantly that we make the system more understandable and flexible. The relational model does not provide a good framework for generalization as it does not "understand" the concepts of "objects" and "inheritance". For this reason, there are some "clashes" between the object oriented thinking promoted by many programming languages and the relational model implemented in many databases. It is possible to connect these two models, but always at the expenses of complex and non-intuitive solutions that often involve a lot of code. On section 4.2 we already discussed some alternatives to the relational model, that are more "friendly" for object oriented programmers.

4.5 Use of Free and Open Source Software

Free and open source software (FOSS) is software that is both *free* and *open-source.* Nowadays the two software movements have drifted apart, so it is important to use a full definition. According to the Free Software Foundation [5](FSF), a "free" software must be licensed to guarantee four types of freedom: *freedom to use, study, share and*

[5] `http://www.fsf.org/`

modify that software. In this definition "free" is not synnonim of "zero cost", although it may be distributed with no charges. An open source license makes source code available to the general public with relaxed or non-existent copyright restrictions.

Apart from ethical reasons which are mainly connected to the "free software movement" (see [42]), there are many motivations to switch to a free and open-source model. Namely:

- **Lower software costs**: Normally FOSS is free and may charge only for support or documentation. If we develop strictly using tools and libraries that follow this model, developing countries will not have to fight with budget restrictions to use, maintain and develop the software. In this way we increase the possibility of ownership by recipient countries.
- **Lower hardware costs**: Generally, FOSS is compact and portable, and as a result it requires less hardware power to accomplish the same tasks as on conventional servers (Windows, Solaris) or workstations. Thus it is still possible to use less expensive or older hardware, allowing to install it in countries with less resources.
- **Simplified license management**: You may obtain as many licenses as you want, to install everywhere, which means that productivity will not be affected by licensing issues.
- **Wide support**: Support for FOSS is often superior to proprietary solutions. This is because we have two levels of support: the free one, that is available through the online community and the paid one, that many tech companies now provide (e.g.: Novell).
- **Software quality**: The peer review process and community standards, plus the fact that source code is disclosed to everyone, tend to drive excellence in design and efficiency in coding.
- **Extended lifetime**: The availability of the source code and the right to modify it enables the unlimited improvement of the software. It also enables to port it to a new hardware or operative system. In reality, no binary-only applications survived more than 10 years in an unmodified form, while many open software systems have survived more than 30 years.
- **Frequent updates**: You do not need to be a programmer to benefit from the availability of the source code; these freedom attracts a community of developers that fixes bugs and develops new features; the right to distribute the modified versions allows the community of users to benefit from all these improvements.
- **Growing community**: The right to use the software in any way, combined with redistribution rights, may attract a large population of users, which helps in turn to build up a market for support and customization of the software, which will attract even more developers to work in the project. If the software is useful, there

is a feedback which will drive the quality of the product, improve its functionality and its support.

There are many problems linked to the use of traditional software, that can be solved using FOSS.

- **No one can restrict in a unilateral way how the software is used (even older versions)**: Sometimes a proprietary software vendor decides to discontinue a software product for an older platform (e.g. Windows 95); in this case the users can either stick to the old version not benefiting from new updates, or switch to another product. In an FOS model, the costumers can keep developing for the desired platform, or alternatively fund somebody to do it.

- **The future of the software is not dependent on a single entity**: With the recent merges in the proprietary software market that led to the "cannibalization" of some software, it is not uncommon that a company decides to completely discontinue a product. In this case, no one is able to prevent the "killing" of the software. In a FOS model, if a company decides to stop development it is always possible to find another alternative, even if it means funding another group to do it.

- **There are no "black boxes"**: By having the source code available, it is possible to assess the correctness of the algorithms and of the implementation scheme used. In the case of scientific applications this may be a very important point, since the quality of the results depends on the correctness of the algorithms. FOSS also grants people the freedom to modify (and share) the code, and therefore they may correct or improve an algorithm, if there is such need.

- **It provides a new forum for democratic action**: As individuals, not companies, decide the overall direction of progress and there are no market forces behind it, there is a more democratic model that allows people with opinions (even if they do not have financial resources) to participate in the improvements of the system.

There are also some arguments against FOSS, some of which are "myths", or are no longer true.

- **There is no support available**: some companies/organizations demand a support model, similar to the one provided by proprietary software. This support is currently available at the same fees, or lower of proprietary software support (e.g. Novell).

- **Development resources are scarce**: Linux and open source are actually the largest segment of the developers universe. Most of the developers tools, languages and processes have a free and open source port. For instance the .NET frame has been implemented with Mono [6].

[6] http://www.mono-project.com/Main_Page

- **There is no training available**: this used to be true, but not anymore. Many companies nowadays provide training and multiple levels of certification in Free and Open Source technologies [7].
- **All open source is a work-in-progress**: this may be true for some software but not all. Although FOSS is evolving constantly, we already see some stable, secure and quality solutions (e.g. Apache, PostgreSQL, QGIS). Although some projects are still maturing, many others are already "usable" as they are.
- **It is difficult to use**: this also used to be true, but not anymore. Initially, FOSS required some advanced knowledge of operative systems (and sometimes programming), in order to be used. Nowadays, products like Ubuntu or QGIS are ready to be installed and used almost by everyone. This is both because they reached some maturity that has driven them to be stable and less "buggy", and a larger community pushed for products to become more "user friendly".

It is true that the users of FOSS are driven to be more "autonomous", since the development model is focused on individuals and communities. It is not true that FOSS is "for developers only", since as we already pointed out, everyone can benefit from involving a large community of developers.

Moreover since they have a wider technology choice, FOSS developers are more exposed to different frameworks/libraries and will be **more likely to try new things**, than their proprietary counterparts. That fact alone does not make them "better" programmers, but makes them at least more adaptable. Technology choices should be ultimately **driven by the nature of the problem, and not by financial/market constraints or by deficiencies of the developers**.

On the context of fishery data management systems, there are a lot of FOS libraries, databases, frameworks and GIS resources that can support the development of the system, in all fronts. On section 2.1 we verified that many systems have chosen this approach, and CAS Medfisis [11] is an example of a fishery data management system that was built *almost entirely* on FOS tools. On the next few paragraphs, we list examples of FOSS that covers different areas involved in the implementation of the proposed system.

- Databases: MongoDB, SQLite, Postgres.
- Spatial Extensions for Databases: SpatiaLite, PostGIS.
- GIS: QGIS, GRASS, SpatiaLite GIS.
- Statistics: R.
- Developing Frameworks: Qt.
- Support Libraries: Boost, STL, PyQGIS, GDAL.
- Versioning System: Git.

[7] for instance, Faunalia - `http://www.faunalia.pt/`

- Wikipedia Server: Mediawiki
- Mockups: Pencil
- UML: Dia
- Documentation: Doxygen
- GeoWebservers: Mapserver, Geoserver, Mapnik.
- Geoweb client: OpenLayers

Note that the all software mentioned above are mature tools, that have been effectively adopted by many projects.

5 Aknoweledgments

I would like to thank Henrik Degel, from the Fishframe project, for kindly providing me with a login that allowed me to try the Fishframe system; Carlos Pinto, from ICES, for amending and "filling the gaps" for the ICES systems reviewed on section 2.1; and Fabio Carocci, from FAO, for suggesting me to review a proprietary GIS fishery system. Finally, I would like to thank the agile-spain-barcelona google group, for removing some of my doubts about AGILE/SCRUM, and for pointing me to valuable reference items on this area.

References

[1] F. Degel, T. Jansen *FishFrame Fisheries and stock assessment data framework.* ICES CM 2006/M:02 , 2006.

[2] F. Degel, T. Jansen *FishFrame Licensing.*Danish Institute for Fisheries Research Charlottenlund castle, Charlottenlund, Denmark. 5th March, 2006.

[3] H. Kjems-Nielsen, L. Inger Larsen, M. Zarecki1, T. Jansen, B. Cowan, P. Sandbeck, M. Dueholm, O. Skov.*InterCatch - a tool for fish stock assessment, status and methods.* ICES CM 2006/M:29, 2006.

[4] ICES. *InterCatch - User Manual.*. Document version 1.9.

[5] ICES *EcoSystemData3.0_Release.* `http://ecosystemdata.ices.dk/EcoSystemData3.0_Release.pdf`

[6] M. Stocker, H. Stiff, W. Shaw, A.W. Argue *The Canadian Albacore Tuna Catch and Effort Relational Database.* Canadian Technical Report of Fisheries and Aquatic Sciences 2701, 2007.

[7] Mappamond *Fishery_Analyst_V2_manual.* `http://www.mappamondogis.it/pdf/Fishery_Analyst_V2_manual.pdf`

[8] L.I. Muhoozi *Implementation of a Fisheries Management Plan (IFMP) Project for Lake Victoria: A Report Of The Fisheries Catch Assessment Survey In The Ugandan Waters Of Lake Victoria For The February 2008 Survey.* National Fisheries Resources Research Institute (NaFIRRI) and National Agricultural Research Organization (NARO), February, 2008.

[9] P. A. Medley *Review Of The Data Collection And Management Systems Of The Marine Fisheries In The Dominican Republican: Final Report.*

[10] N. Helm *A Report on The Market Survey of Reef and Lagoon Fish Catches In Western Samoa.* South Pacific Comission, SPC/lnshoreFish.Res./BP 30, 7 March 1988.

[11] FAO.*Medfisis*,`http://www.faomedfisis.org/`.

[12] JSON.*JSON*,`http://json.org/`.

[13] SQLite.*SQlite*,`http://www.sqlite.org/`.

[14] H.H. Shahri, A.A.Z. Barforush.*Data mining for removing fuzzy duplicates using fuzzy inference.* Fuzzy Information, 2004. Processing NAFIPS '04. IEEE Annual Meeting, 27-30 June 2004. `http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1336319`

[15] C. López. *Quality of Geographic Data Detection of Outliers and Imputation of Missing Values.* PhD Dissertation on the Royal Institute of Technology, Department of Geodesy and Photogrammetry, Stockholm, Sweden 1997.

[16] ICES. *Workshop on methods for merging metiers for fishery based sampling (WKMERGE)* . ICES WKMERGE REPORT 2010, Copenhagen, Denmark, 19-22 January 2010.

[17] R. Forrest, T. Pitcher, R. Watson, H. Valtýsson, S. Guénette *Estimating Illegal and Unreported Catches from Marine Ecosystems: Two Case Studies.* Sea Around Us: North Atlantic, Page 81, 19 DEC 2002.

[18] ICES. *ICES Data Centre*, `http://www.ices.dk/datacentre/Submissions/index.aspx`.

[19] J. Simoes, C. Pinto, M. Afonso-Dias. *Methodology for Monitoring and Management of the Crustacean Trawl Fleet. Example of GeoCrust 1.0 GIS* in Finisterra special edition Cartography and Geographic Information Systems, 2005.

[20] D. Z. Sui *A Fuzzy Gis Modeling Approach Land Evaluation For Urban.* Environ. and Urban Systems, Vol. 16, pp. IOl-115, USA, 1992.

[21] W. Kainz. *The mathematics of GIS.* `http://goo.gl/fb/CWklR`

[22] FAO/DANIDA. *Guidelines for the Routine Collection of Capture Fishery Data.* FAO FISHERIES TECHNICAL PAPER 382. Food and Agriculture Organization of the United Nations. Rome, 1999.

[23] *fisheries_data_mng_sys.sqlite.*

[24] L. R. Garces, G. T. Silvestre, I. Stobutzki, F. C. Gayanilo Jr, F. Valdez, M. Saupi, T. Boonvanich, M. Roongratri, P. Thouc, Purwanto, I. Haroon, K. N. Kurup, M. Srinath, H. A. B. Rodrigo, M. D. Santos, F. S. B. Torres Jr., M. K. Tan, D. Pauly *A regional database management system—the fisheries resource information system and tools (FiRST): Its design, utility and future directions.* Fisheries Research, 78 (1-2). pp. 119-129.

[25] IIP. *PESCART: Documentacao tecnica.* 2006.

[26] European Comission. *Common tool for raising and estimating properties of statistical estimates derived from the Data Collection Regulation (COST).* Studies and Pilot projects for carrying out the common fisheries policy. Call for proposal FISH/2006/15 – lot 2.

[27] COST Team and various contributors. *Package 'COSTcore' documentation.* February, 15, 2008.

[28] B. U. Ibrahim, J. Auta, J. K. Balogun,*A Survey Of The Artisanal Fisheries Of Kontagora Reservoir, Niger State, Nigeria.* Bayero Journal of Pure and Applied Sciences, 2(1): 47 - 51. 2009.

[29] Uncle Bob. *NO DB.* `http://blog.8thlight.com/uncle-bob/2012/05/15/NODB.html`

[30] Gatherspace. *Writing Effective Use Case Examples.* `http://www.gatherspace.com/static/use_case_example.html`

[31] Wikipedia. *Relational model* `https://en.wikipedia.org/wiki/Relational_model`

[32] Wikipedia. *NoSQL* `https://en.wikipedia.org/wiki/NoSQL`

[33] Wikipedia. *Eventual Consistency* `http://en.wikipedia.org/wiki/Eventual_consistency`

[34] Wikipedia. *CAP* `https://en.wikipedia.org/wiki/CAP_theorem`

[35] AGILE. *Manifesto* `http://www.agilemanifesto.org/`

[36] GISFOREAF. *Project Wiki Page* `http://ladybug.no-ip.org:8010/mediawiki/index.php/GISforEAF`

[37] Ambysoft. *Roles on Agile Teams: From Small to Large Teams* `http://www.ambysoft.com/essays/agileRoles.html`

[38] Wikipedia. *SCRUM* `http://en.wikipedia.org/wiki/Scrum_(development)`

[39] K. Schwaber, J. Sutherland*The Scrum Guide. The Definitive Guide to Scrum: The Rules of the Game.* 1991-2011.

[40] P. Deemer, G. Benefield, C. Larman, B. Vodde *INFORMACIÓN BÁSICA DE SCRUM (THE SCRUM PRIMER).* Version 1.1. SCRUM Training Institute.

[41] H. Kniberg*SCRUM Y XP DESDE LAS TRINCHERAS Cómo hacemos Scrum.* Version 1.1. Enterprise Software Development Series.

[42] S. Williams*Free as in Freedom.* 2002. Project Gutenberg.