Deliverable No. 3.4

Project acronym: EcoFishMan

Project title: "Ecosystem-based Responsive Fisheries Management in Europe"

Grant agreement No: FP7-265401

Project co-funded by the European Commission within the Seventh Framework Programme

Start date of project: 1st March 2011

Duration: 36 months

Due date of deliverable: Originally M24, Revised M36
Submission date: M36
File Name: D3 4 EcoFishMan Final version of GIS-based support system
Revision number: 01
Document status: Final
Dissemination Level: PP

Revision Control

| Role                  | Name                  | Organisation | Date    | File suffix
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Authors</td>
<td>J Spicer</td>
<td>Mapix</td>
<td>27/02/14</td>
<td>JS</td>
</tr>
<tr>
<td></td>
<td>Sigridur Sigurdardottir</td>
<td>Matis</td>
<td>05/02/14</td>
<td>SS</td>
</tr>
<tr>
<td></td>
<td>Gert Riomersma</td>
<td>Mapix</td>
<td>28/02/14</td>
<td>GR</td>
</tr>
<tr>
<td></td>
<td>Petter Olsen</td>
<td>Nofima</td>
<td>28/02/14</td>
<td>PO</td>
</tr>
<tr>
<td>WP leader</td>
<td>Petter Olsen</td>
<td>Nofima</td>
<td>28/02/14</td>
<td>PO</td>
</tr>
<tr>
<td>Administrative Officer</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Coordinator</td>
<td>Anna Kristín Danielsdóttir</td>
<td>Matis</td>
<td>28/02/14</td>
<td>AKD</td>
</tr>
</tbody>
</table>

1 Document will be a draft until it was approved by the coordinator
2 PU: Public, 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)
3 The initials of the revising individual in capital letters
Contents

Figure overview ........................................................................................................... 2
Executive summary ......................................................................................................... 3
Background .................................................................................................................... 4
Icelandic fisheries management ..................................................................................... 5
The simulation model ..................................................................................................... 5
Initial R Model ............................................................................................................... 7
Evolution of the Model ................................................................................................ 9
Final Output from the Web based model .................................................................... 13
Technical details of the Web based model .................................................................. 14
Appendix A – R-Model code ....................................................................................... 16
Appendix B – Web browser code .................................................................................. 23
Appendix C – OpenLayers code ..................................................................................... 30

Figure overview

Figure 1: Parameters in growth and harvest function .................................................. 6
Figure 2: Initial R Model ............................................................................................. 7
Figure 3: Graphical output from R-Model ..................................................................... 8
Figure 4: Textual output from R-Model ........................................................................ 8
Figure 5: R-Model interface to web server ................................................................... 9
Figure 6: R-Model web browser display ...................................................................... 10
Figure 7: Web browser input dialogue ......................................................................... 10
Figure 8: HTML code for web browser interface ....................................................... 11
Figure 9: getHarbours code to return harbour map data ............................................ 12
Figure 10: Web interface showing interactive map with graph minimized ................. 13
Figure 11: Web interface showing interactive map with graph maximised ............... 13
Figure 12: Diagram details the data-flow when running the web model .................... 14
Executive summary
This is deliverable report D3.4 in the EcoFishMan project documenting the GIS application that was developed to enable simulation and simple “what-if” analysis for the Icelandic case. The Icelandic case was chosen because simulation of this type requires a lot of data on very detailed level, and the Icelandic case was most mature in this respect. The simulation lets the user play around with the ratio of ITQ quotas over hood and line quotas, and the effect on the stock is visualized in different Icelandic regions.

This report outlines the data, the code and provides some screenshots to indicate how the application looks to the user. Note that this is a report documenting the deliverable; the actual deliverable is the application which is openly available on the internet and can be run and tested by anyone.
**Background**

D3.4 is the last formal deliverable from EcoFishMan WP3. The WP3 objectives that pertain to D3.4 are the following:

- Develop a web based decision support management tool
- Deploy the tool using the data and models developed for the various case studies.

Specifically, D3.4 creates a prototype “decision support Fisheries management tool” for the EcoFishMan project based on the Case Studies. Specifically it uses, as the prototype, the fisheries Model developed for Icelandic demersal catches and integrates it with a web-based application which is map centric. Functionality for displaying data from the case studies and visualizing the geographical and temporal aspect of the data has been developed and documented previously in EcoFishMan. See deliverable D3.1, D3.2 and D3.3 for detailed description of this functionality and for the interactive maps developed for the respective case studies.

This application is a prototype and was due to be finished by M36, to be documented in D3.4.

Note that the web-based application itself, collated, harmonized and with all the data from the Icelandic case, is the actual deliverable. This is the deliverable report, mainly containing screenshots to show how the data looked before collection, and how the database looks after collation and harmonization.

The website which runs the application is hosted by MAPIX technologies and is available at the following url:

[http://ecofishman.mapix.com](http://ecofishman.mapix.com)
Icelandic fisheries management

Management of demersal catches in Iceland is divided into three segments:

1. ITQ (Individual Transferable Quota System) where vessels are allocated quotas according to their permanent quota shares.
2. A hook system where vessels are allocated quotas according to their quota shares. Only vessels less than 15 meters in length and use hand line or longline are qualified for this system.
3. Coastal fisheries where vessels smaller than 15 meters using hand line can catch demersal species in an “Olympic” fishery in May-August each year.

The simulation model

The model, which only includes the cod fishery, looks at the ITQ system and the hook system which together account for nearly all of the total allowable catch for cod.

The main objective of the model is to show how the ratio between the ITQ quotas (often referred to as the “big system”) and the quotas in the hook and line system can vary over time throughout the country.

The population dynamics of the are described with a logistic growth model:

\[ x(t+1) - x(t) = r x(t) (1 - x(t)/K) - h(t) \quad (1) \]

where \( x(t) \) is the biomass of fish at year \( t \), \( r \) is the intrinsic growth, \( K \) the carrying capacity and \( h(t) \) the harvest at year \( t \). It is assumed that harvest, or total allowable catch (TAC) is determined by harvest control rules:

\[ h(t+1) = F x(t+1) = TAC(t+1) = a x(t) + TAC(t) \quad (2) \]

---

where \( F \) is fishing mortality and \( a \) is harvest rate. Spawning stock biomass (SSB) is assumed to be a certain ratio, \( r_{SSB} \) of biomass:

\[
SSB_t = r_{SSB} x_t \quad (3)
\]

where \( r_{SSB} \) is a uniformly distributed random variable on an interval which is obtained from analysis of SSB data from the Icelandic Marine Research Institute.

The biological model was fitted using data from the Icelandic Marine Research Institute\(^5\) and is found in Table 1.

<table>
<thead>
<tr>
<th>Species</th>
<th>Function</th>
<th>Parameters</th>
<th>t-statistics</th>
</tr>
</thead>
</table>
| Cod     | \( r_{xt} - xtK - h_t \)  
          | \( h_t = aX_t + TACt^2 \)  
          | \( SSB_t = r_{SSB} x_t \) | \( r = 0.470 \)  
          | \( k = 2654 \)  
          | \( a = 0.2 \)  
          | \( r_{SSB} \sim U(0.24;0.39) \) | \( r: 6.65 \)  
          | \( K: 2.56 \) |

**Figure 1:** Parameters in growth and harvest function

The allocation scenarios calculations are based on the most recent quota allocation data obtained from the Directorate of fisheries\(^6\).


\(^6\) [http://www.fiskistofa.is/veidar/aflaheimildir/uthlutadaflamark/](http://www.fiskistofa.is/veidar/aflaheimildir/uthlutadaflamark/)
Initial R Model

The initial R model had a basic input set by variables in the script to produce a basic graph and data output.

```r
# Hardcoded parameters
r11 = 0.515 # ratio of Big system quota that goes to trawlers
r12 = 0.454 # ratio of Bigsystem quota that goes to other vessels
r13 = 0.031 # ratio of Bigsystem quota that goes to small vessels
# Cost/revenue - data obtained from Statistics Iceland (average of last 3 years)
c11 = 0.779 # trawlers
c12 = 0.722 # other vessels
c13 = 0.761 # small vessels
c2 = 0.695 # hook and line

# Input parameters
# Fish price (how much a vessel gets per kg)
p1 = 280 # price of cod
a_cod = 0.2 # catch rate
r1 = 0.834 # ratio of quota to big system
r2 = 1.1 # ratio of quota to hook and line system

# Set up time increment and vector of time values
deltaT = 1 # year
t = seq(0, time, deltaT) # simulation time

# Population dynamics
K_cod = 2654000000 # carrying capacity
initTAC_cod = 644000000 # initial biomass
initTAC_cod = 272000000 # initial TAC
```

**Figure 2: Initial R Model**

As seen above a set of input variables are defined in the model, some of these parameters are then adjusted and the model is re-run to obtain the updated result output. The model outputs a graph based on these figures alongside a set of results that is returned in text format.
Figure 3: Graphical output from R-Model

![Graphical output from R-Model](image.png)

Figure 4: Textual output from R-Model

```
# Textual output from R-Model
```

![Textual output from R-Model](image.png)
Evolution of the Model

The initial step in moving the model to the web was to move the model into a container that allows the user with their web browser to change the values of variables and recalculate the output. The Shiny application was used to achieve this, this enables R-Models to be run on a server with the output sent to a web browser.

The information from the shiny website (http://www.rstudio.com/shiny/) is as follows:

"Shiny makes it super simple for R users like you to turn analyses into interactive web applications that anyone can use. Let your users choose input parameters using friendly controls like sliders, drop-downs, and text fields. Easily incorporate any number of outputs like plots, tables, and summaries. No HTML or JavaScript knowledge is necessary. If you have some experience with R, you’re just minutes away from combining the statistical power of R with the simplicity of a web page."

"While the Shiny package itself includes a basic web server, it’s only designed to serve one application at a time. And if an error occurs that causes the Shiny application to exit, your application will be down until you manually start it up again. Shiny Server is designed to serve up multiple Shiny applications on the same server. It handles the job of launching and managing the R processes that run your Shiny apps, and directs web traffic to the correct process based on request URL."

```
shinyServer(function(input, output) {

  runSim <- function() { stockSim(input$sim_time, input$initXcod, input$initTxCod, input$sa_cod) }
  output$codPlot <- renderPlot({
    attach(runSim())
    plot(a, type="l", xlab="Time (Years)", ylim=c(min(a), max(a)))
    lines(b, type="line", col="red")
    abline(h = 0, lty = FALSE, col="blue")
  })
})

stockSim <- function(time = 1, initXcod = 994006800, initTxCod = 192000680, a_cod = 1) {
  #Set up time increment and vector of time values
```

Figure 5: R-Model interface to web server
This Shiny server interface was then followed by implementing additional code to export the model to a web format as well as taking input from the User, interacting with their web browser, and allowing execution of the model with the external inputs.

Simulation of cod, haddock population in Icelandic waters

Figure 6: R-Model web browser display

Once the ability to update inputs to the model was in place, the initial work then began on the web based interface, creating an initial input dialogue as seen above with some requested inputs. The output graph was then added to this page to give basic feedback and output from the model.

Figure 7: Web browser input dialogue
Having tested the basic web interface, further development work was then undertaken to create and layout the web map interface. Drawing on the previous developed web based interactive fisheries maps (e.g. Icelandic Case Study for the Ecofishman project), controls for data input and graph output were overlayed onto the map interface.

**Figure 8:** HTML code for web browser interface
Figure 9: getHarbours code to return harbour map data

After having developed the basic web interactive map, development was then undertaken to create the background functions to gather the data required for adding the Harbours and calculated harbour landing data to the map.

These functions were then complemented with an array of functions for merging the computed values for each Harbour, obtained from the R-Model together with the location data of the harbours into an array. This array was then used to display the harbours, complete with R-model output, onto the interactive map.

The mapping used to overlay the harbours on was Openstreet map. For more information please see http://www.openstreetmap.org/about

The final additions to the web interface were:

a) an additional parameter “R1 - Ratio of quota to big system” that could be varied and used directly as an input to the R-model

b) A button to give the user the ability to hide and display the graph. The graph displayed on the web interface is “Spawning stock biomass of cod"
Final Output from the Web based model

**Figure 10:** Web interface showing interactive map with graph minimized

**Figure 11:** Web interface showing interactive map with graph maximised
Technical details of the Web based model

The Web Model consists of several components.

The main simulation model is written in R, this in turn makes use of an ODBC connection to collect data from a database as well as buffering a CSV file with data required for running the simulation.

The model is triggered from the web via Shiny, a server application that allows data to pass from a Web based interface through to an R-Model instance.

The Front end component is an HTML and Javascript webpage that makes use of the Jquery and Openlayers libraries in order to display the data on a map component.

Figure 12: Diagram details the data-flow when running the web model
When the model is initially loaded into the web browser the shiny server runs the R-model with the initial input values. The R model then queries the Harbour data in the SQL database and returns it via the Shiny server to the Map interface. The model proceeds to load the required data from the CSV and run the simulation. Finally it returns the result set to the Shiny server and from there is directed to the Web browser interface and the Openlayers Map Interface.

The Web browser interface makes use of Javascript functions to update the Shiny layer with current input values. These are then used in the simulation model and the results are returned both in the form of an updated graph image and result data which is then passed to the Openlayers Javascript library and used to create the map element of the display.
Appendix A – R-Model code

R-Model embedded within the Shiny Web Server

Software code
# Simulation of cod, haddock population in Icelandic waters

# Include the shiny library for the web functions

library(shiny)

library("TTR")

library("forecast")

library("RODBC")

shinyServer(function(input, output, session) {
  # Output Harbour details for map layer
  output$harbours <- renderTable({getHarbours()})

  harbourinfo = getHarbours()

  # Setup function call for shiny with the inputs from the web front end
  runSim <- function() {
    stockSim(input$time,input$initXcod,input$initTACcod,input$a_cod,input$r1)
  }

  # Create the output graph that redraws when inputs change.
  output$codPlot <- renderPlot({
    # Get outputs from the stockSim
    attach(runSim())

    # Plot OT_1
    plot(a,type="l",xlab="Time (Years)",ylab="Kilotons",ylim=c(min(a),max(a)))

    # Plot line at OT1_lim
    lines(b,type="line",col="red")

    # Draw blue line at 0 if data goes below 0
    abline(h = 0,untf = FALSE,col="blue")

    title(main="Spawning stock biomass of cod")
  })
})
output$codTbl <- renderTable{
    attach(runSim())
    data.frame(results)
}
output$harbourTbl <- renderTable{
    attach(runSim())
    data.frame(harbours)
}
observe{
    attach(runSim())
    harbours
    session$sendCustomMessage(type = "resultsCallbackHandler",
                              data.frame(harbours) )
}
observe{
    harbourinfo
    session$sendCustomMessage(type = "infoCallbackHandler",
                              data.frame(harbourinfo) )
}
}

getHarbours <- function() {
    ## get harbour data from mysql database
    dbhandle <- odbcDriverConnect('driver=FreeTDS;server=109.233.115.215;port=1433;database=EcoFishMan;uid=EcoFish;tds_version=8.0')
    ##!!! MUST SET THESE DATABASE DETAILS, OTHERWISE IT NO WORKY!!
    res <- sqlQuery(dbhandle, "select HarbourID, Harbour_Na, Lat, Long, Admin, Geom.STAsText() as Geom from Icelandic_Harbours where HarbourID < 151")
    #data <- fetch(res, n = -1)
close(dbhandle)
return(res)
}

#define function for running the simulation with input variables and default values
stockSim <- function(time = 1, initXcod = 944000000, initTAC_cod = 172000000, a_cod = 0.2, r1 = 0.824) {

    # model follows pretty much unchanged, some variables moved into the function above.

    #Read quota information
    ## Column information: harbour, BigSystem, HookLine, Vessels, SmallVessels ##
data_cod = read.csv2("Cod_ratio.csv", header = TRUE, sep=";",dec = ".")

    ###########################Hardcoded parameters#################
    #### At least in the first version of the DSS, we should keep the following
parameters hardcoded ######
    r11=0.511 #ratio of BigSystem quota that goes to trawlers
    r12=0.464 #ratio of BigSystem quota that goes to other vessels
    r13=0.026 #ratio of BigSystem quota that goes to small vessels

    ###Cost/revenue - data obtained from Statistics Iceland (average of last 3 years)
    ####
    c11 = 0.779 #trawlers
    c12 = 0.722 #other vessels
    c13 = 0.766 #small vessels
    c2 = 0.805 #hook and line

    ###########################Input parameters####################
#Fish price (how much a vessel gets per kg)#

\[ p_1 = 280 \] #price of cod

#Catch rate
\[ a_{cod} = 0.2 \]

#Ratio of quota to big system
\[ r_1 = 0.824 \] ##ratio of quota to big system

#Derived value
\[ r_2 = 1 - r_1 \] ##ratio of quota to hook and line system

#Set up time increment and vector of time values

\[ \text{time} = 10 \]

\[ \text{deltaT} = 1 \text{ year} \]

\[ t = \text{seq}(0, \text{time}, \text{deltaT}) \] #simulation time

#Constants cod #
\[ r_{cod} = 0.470 \] #growth rate

\[ K_{cod} = 2654000000 \] #carrying capacity

#Stock variables and initial conditions for COD 

\[ X_{cod} = \text{vector(length=length(t))} \]

\[ X_{cod}[1] = \text{initXcod} \]

\[ TAC_{cod} = \text{vector(length=length(t))} \]

\[ TAC_{cod}[1] = \text{initTAC_cod} \]

for (i in 2:length(t)) {

\[ \text{compute rates of change} \]

\[ \text{BioMcod} = r_{cod}*X_{cod}[i-1]*(1-X_{cod}[i-1])/K_{cod} \]
harvest_cod = (a_cod*Xcod[i-1]+TAC_cod[i-1])/2

# increase/decrease the population
Xcod[i] = Xcod[i-1] + (BioMcod) * deltaT - (harvest_cod) * deltaT
TAC_cod[i] = harvest_cod

### Quota calculations ###
q11=vector(length=length(t))
q12=vector(length=length(t))
q13=vector(length=length(t))
q2=vector(length=length(t))

for (i in 1:length(t)) {
    q11[i]=TAC_cod[i]*r11*r1
    q12[i] = TAC_cod[i]*r12*r1
    q13[i] = TAC_cod[i]*r13*r1
    q2[i] = TAC_cod[i]*r2
}

Vessels=as.numeric(data_cod$Vessels)

### Calculate quota for each harbour next n years ###
Quota_Trawlers=data_cod$Trawlers*q11
Quota_Vessels=Vessels*q12
Quota_SmallVessels=data_cod$SmallVessels*q13
Quota_HookLine = data_cod$HookLine * q2

### Results to show on map #######

Results_Quota = Quota_Trawlers + Quota_Vessels + Quota_SmallVessels + Quota_HookLine

Harbour_Data = list(Harbour = data_cod$Harbour, DBID = data_cod$DBID, results = Results_Quota)

SSB_B4_cod = runif(length(t), 0.24, 0.39)

OT_1 = SSB_B4_cod * Xcod

OT_1 = round(OT_1 / 10^6)

OT1_lim = rep(220, length(t))

# return the results for use outside the function
return(list(a = OT_1, b = OT1_lim, results = Results_Quota, harbours = Harbour_Data))

# plot results
# now run inside shinyServer block

# attach(stockSim(time=10,a_cod=1)) added to test function in rstudio.
# plot(a,type="l",ylim=c(min(a),max(a)))
# lines(b,type="line",col="red")
Appendix B – Web browser code

Web Browser Index Page

Software Code
Simulation of cod, haddock population in Icelandic waters

Timescale (years)

10
<table>
<thead>
<tr>
<th>Label</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>initXcod</td>
<td>Initial biomass of cod (kg)</td>
<td>944000000</td>
</tr>
<tr>
<td>initTACcod</td>
<td>Harvest in first year of simulation (kg)</td>
<td>172000000</td>
</tr>
<tr>
<td>a_cod</td>
<td>Harvesting Policy</td>
<td>0.2</td>
</tr>
<tr>
<td>r1</td>
<td>Ratio of quota to big system</td>
<td>0.824</td>
</tr>
</tbody>
</table>

---

**Show Graph**:<br>
![chart](icons/chart_line.png)
```javascript
$('#codPlot').hide();
$('#codPlotWrapper').css('width', '20px');
$('#codPlotWrapper').css('height', '20px');
$('#codPlotSizeToggle').html('[+]');

var resultsreturned = null;
var inforeturned = null;

Shiny.addCustomMessageHandler("resultsCallbackHandler",
    function(harbourresults) {
        resultsreturned = harbourresults;
        //alert("got new results");
        if (resultsreturned != null && inforeturned != null)
        sortOutData();
    }
);

Shiny.addCustomMessageHandler("infoCallbackHandler",
    function(harbourinfo) {
        inforeturned = harbourinfo;
        //alert("got new harbour info");
        if (resultsreturned != null && inforeturned != null)
        sortOutData();
    }
);

function sortOutData () {
    var ir = inforeturned;
```
harbourLayer.destroyFeatures();

for (var i = 0; i < ir.HarbourID.length; i++) {
    ID = ir.HarbourID[i];
    Name = ir.Harbour_Na[i];
    Lat = ir.Lat[i];
    Long = ir.Long[i];
    Admin = ir.Admin[i];
    Geom = ir.Geom[i];
    Result = findByHarbourID(ID);
    Radius = Result / 350000.0;
    harbour_geom = new OpenLayers.Geometry.fromWKT(Geom);
    harbour_geom.transform(wgs84, sph_mer);
    harbour_point = new OpenLayers.Feature.Vector(harbour_geom);
    harbour_point.attributes = {
        ID: ID,
        Name: Name,
        Lat: Lat,
        Long: Long,
        Admin: Admin,
        Result: Result,
        Radius: Radius
    };
    harbourLayer.addFeatures([harbour_point]);
}

harbourLayer.redraw();
function findByHarbourID(id) {
    var rr = resultsreturned;
    for (var i = 0; i < rr.Harbour.length; i++) {
        if (rr.DBID[i] == id) {
            return rr.results[i];
        }
    }
    return 0;
}

</script>
</body>
</html>
Appendix C – OpenLayers code

Binding Shiny with OpenLayers

Software Code
// This output binding handles statusOutputBindings

var harbourOutputBinding = new Shiny.OutputBinding();
$.extend(harbourOutputBinding, {
    find: function(scope) {
        return scope.find('.harbour');
    },
    renderValue: function(el, data) {
        jQuery(data).find("entry").each(function() {
            ID = entry.attr("ID");
            Name = entry.attr("Name");
            Lat = entry.attr("Lat");
            Long = entry.attr("Long");
            Admin = entry.attr("Admin");
            Geom = entry.attr("Geom");
            harbour_geom = new OpenLayers.Geometry.fromWKT(Geom);
            harbour_geom.transform(wgs84, sph_mer);
            harbour_point = new OpenLayers.Feature.Vector(harbour_geom);
            harbour_point.attributes = {
                ID: ID,
                Name: Name,
                Lat: Lat,
                Long: Long,
                Admin: Admin
            };
        });
    }
});
harbourLayer.addFeatures([harbour_point]);
}

map.addLayers([harbourLayer]);

Shiny.outputBindings.register(harbourOutputBinding, 'harbour.harbourOutputBinding');